A Shapley-érték komplexitása és becslése

Doktori értekezés

Írta: Illés Ferenc

Közgazdasági és Gazdaságinformatikai Doktori Iskola

Témavezető:

Dr. Csóka Péter

Befektetések Tanszék

Budapesti Corvinus Egyetem, Pénzügy Intézet

Budapesti Corvinus Egyetem

Pénzügy Intézet

2023

Tartalomjegyzék

Bevezetés

A kooperatív játékelmélet számos társadalmi dilemma illetve pénzügyi és gazdasági probléma modellje. Általában akkor alkalmazzuk, ha egy közösség által elérhető eredmény meghaladja az egyénenként elérhető eredmények összegét (szinergia), vagy egy tevékenység teljes költsége alacsonyabb, mint a részfeladatok költségeinek összege (megtakarítás) illetve teljes kockázata kisebb, mint a részfeladatok aggregált kockázata (diverzifikáció). A fő kérdés pedig természetesen az, hogy hogyan osszuk el a többletet a szereplők között valamilyen „igazságos" módon. A lehetséges alkalmazások széles spektrumát illusztrálják a következő példák:

- Az egyik legrégebbi kooperatív játékelméleti probléma az i.sz. 1140-ből származik, és a Talmud-ban található: „Jákob meghalt és elsőszülött fiára hagyta teljes vagyonát, második fiára a felét, harmadikra a harmadát, és legkisebb fiára a negyedét." (Driessen, 1988). A rabbi tehát, nagylelkűen teljes vagyonának 25/12 -ed részét, valamint azt a problémát hagyta gyermekeire, hogy most aztán osszák szét, ami van, ahogy tudják. Mit tegyenek a fiúk?

- Egy kereskedelmi bank treasury osztályán harminc bróker dolgozik, akik a piacon elérhető többezer lehetséges eszközből állíthatnak össze tetszőleges portfóliót. Az osztály teljes költségkerete harminc milliárd forint, a bank aggregált portfóliójának tőkekövetelménye ezt az értéket egyetlen időpillanatban sem haladhatja meg. A tőkekövetelmény a 99,9%-os CVaR értéke[1], melyet a portfóliót alkotó eszközök hozamának historikus adataiból becsült várható értékek és kovarianciamátrixuk alapján számolnak ki, az eszközök hozamának többdimenziós normális eloszlását feltételezve. Az osztályvezető az egyenlőség elvét szem előtt tartva, mindegyik brókernek 1 milliárd forintos keretet hagy jóvá, amit a kollégák külön-külön be is tartanak, azonban, amikor a bank aggregált portfóliójának tőkekövetelményét kiszámolják, az mindössze 13 milliárd forint. A különbözet oka, hogy a tőkekövetelmény, a hozamok közti diverzifikációs hatás miatt nem additív. Ráadásul, az, hogy egy bróker által tartott eszközök hogyan változtatják meg az aggragált portfólió kockázatát, az összes többi eszközzel való korrelációtól is függ. Lehetséges, hogy valaki

[1] *Conditional Value at Risk*, azaz feltételes kockáztatott érték egy gyakran használt kockázati mérték.

Ország	Szavazatok száma	Népesség	
Franciaország	4	44	millió fő
NSZK	4	51	millió fő
Olaszország	4	49	millió fő
Belgium	2	9	millió fő
Hollandia	2	11	millió fő
Luxemburg	1	308	ezer fő

1. táblázat. EGK országok szavazóereje és népessége

önmagában látszólag nagyon-nagy kockázatot vállal, de ez a teljes portfólióra nézve valójában fedezetet képez. Hogyan osszuk szét a portfólió teljes kockázatát az egyes eszközök között? A probléma a szakirodalomban tőkeallokáció vagy kockázatfelosztás (*risk capital allocation*) néven terjedt el (ld. például Colini-Baldeschi et al. (2018), Csóka et al. (2009) illetve Csóka és Pintér (2016)).

- Az Európai Gazdasági Közösség (vagy más néven Közös Piac) nevű nemzetközi megállapodást 1957-ben hat ország írta alá, melyek nem egyenlő súlyozással, többségi szavazással hoztak döntéseket. Egy pozitív döntéshez legalább 12 igen szavazat kellett a 17-ből, melyet a tagországok a 1. táblázatban megadott súlyok alapján kaptak (Taylor és Pacelli, 2008). Összehasonlításképpen a táblázat harmadik oszlopában felüntettük az országok becsült népességét[2]. Azt gondolhatnánk, hogy Luxemburg a népességéhez képest erősen túlreprezentált, hisz negyedakkora a szavazóereje, mint a százötvenszer akkora Nyugat-Németországé, ha azonban kicsit jobban megnézzük a számokat, láthatjuk, hogy a fenti rendszerben nem képzelhető el egyetlen olyan döntési szituáció sem, ahol Luxemburg szavazata számítana. Ez ugyanis csak akkor fordulhatna elő, ha a másik öt ország úgy szakadna két pártra, hogy az adott kérdést támogatók szavazatainak összértéke 11, az ellenzőké pedig 5. Ezt azonban lehetetlen 5 darab páros számból kirakni.

- Lényegesen hétköznapibb probléma a következő. Két utas szeretne taxiköltségen osztozni. Az első utas A-ból megy C-be, a másik utas félúton, B-ben száll be a taxiba. Hányad részét fizesse a teljes költségnek a második utas?

- A kooperatív játékelméletnek olyan területeken is van relevanciája, ahol elsőre nem is gondolnánk. Egy ilyen terület például a jog. Többes károkozásról akkor beszélünk, ha több egymástól függetlenül elkövetett hibából származik valakinek kára, de azt különkülön egyik hiba sem okozta volna, vagy legalábbis sokkal kisebb kárt okoztak volna

[2]forrás: https://www.worldometers.info/population/countriesineuropebypopulation, illetve NSZK esetén https://en.wikipedia.org/wiki/West_Germany

(ld. például Ferey és Dehez (2016) illetve Dehez és Ferey (2013)). Tipikus példa erre a szituációra, ha egy autós elüt egy gyalogost, akinek könnyebb sérülése keletkezik, például eltörik a lába, azonban a kórházban okozott műhiba következtében további maradandó károsodást szenved. Milyen arányban térítse meg a kórház és az autós a kárt?

- Másik meglepőnek tűnő alkalmazásként a kooperatív játékok elmélete a többváltozós statisztikában regressziós modellek magyarázó változóinak értékelésére is használható (Pintér 2007, 2011). Itt a teljes modell magyarázóerejét (például R^2) szeretnénk szétosztani az egyes magyarázó változók között hogy jobban megértsük, melyek a fontos változók.

A kooperatív játékelméletnek a pénzügyek területén is számtalan alkalmazása van. Néhány példa a szakirodalomból:

- A klasszikus Markowitz-féle portfólió-optimalizálási probléma (Markowitz, 1955) feltevései szerint a befektető $w \cdot \mu - A \cdot w^t \cdot \Sigma \cdot w$ alakú hasznosságfüggvényt maximalizál, ahol w a portfóliósúlyokból álló vektor, μ az eszközök várható hozamaiból álló vektor, Σ az eszközök hozamainak kovarianciamátrixa, A pedig egy szubjektív kockázatkerülési paraméter. A probléma egy érdekes általánosítását kaphatjuk, ha figyelembevesszük, hogy a részvények birtoklása a tulajdonosnak kontrolljogokat is biztosít, azaz beleszólhat a cég működésébe, szavazhat a közgyűlésen a menedzsment kinevezéséről/leváltásáról stb. Ha a befektető a cégben megszerzett befolyást, is figyelembeveszi a portfólió kialakítása során, az optimális portfólió megváltozhat. Amihud és Barnea (1974) egy ilyen portfólió-optimalizálási problémát modellez, melyben a cégben megszerzett kontroll számszerűsítésére kooperatív játékelméleti eszközöket használ. Az optimalizálási probléma ebben a kiterjesztett modellben diszkrét jelleget ölt, a Markowitz-féle kvadratikus programozási feladathoz képest sokkal nehezebb.

- A kooperatív játékelmélet a (pénzügyi) piacok modellezésére is nagyon hasznos. Alapvetően fontos közgazdaságtani eredmény, hogy ez egy olyan játékosztály, melyben mindig létrejön a kooperáció (Shapley és Shubik, 1969).

- A 2008-as válság egy korábban ismeretlen kockázatra hívta fel a figyelmet. A rendszerkockázat a teljes pénzügyi szektor összeomlásának, kiszáradásának veszélye, melynek számszerűsítése egyelőre komoly kihívás a kockázatkezelőknek és a pénzügyi intézményeket szabályozó, felügyelő hatóságoknak, és intenzív kutatás tárgyát is képezi. Tarashev et al. (2016) ennek az igen időszerű problémának a modellezésére használja a kooperatív játékelmélet eszközeit.

- Végezetül érdemes megemlíteni egy nem túl szívderítő, de társadalmilag fontos területet:

Chantreuil et al. (2020) jövedelemegyenlőtlenségek vizsgálatára használja a kooperatív játékelméletet.

A kooperatív játékok legismertebb általános megoldását Lloyd Stowell Shapley adta meg 1953-ban *A value for n-person games* című cikkében. Bizonyítható, hogy Shapley megoldása, melyet azóta Shapley-értéknek neveznek, az egyetlen általános, azaz minden játékra működő megoldási koncepció, mely bizonyos igazságossági feltételeknek eleget tesz. Szépséghibája, hogy – az egyszerűen felírható formula és intuitív tartalma ellenére – konkrét játékokra nem feltétlenül tudjuk az értékét pontosan meghatározni, ha a játékosok száma túl nagy, ugyanis a képlet a játékosok számában nem polinomiális számú tag összegéből áll. Ennek a dolgozatnak a tartalma, ha egy mondatban kéne összefoglalni: Hogyan lehet (illetve nem lehet) egy kooperatív játék Shapley-értékét hatékonyen kiszámolni vagy megbecsülni?

A dolgozat felépítését tekintve öt további fejezetből áll. Az első fejezet a kooperatív játékelmélet alapjait, a később használt jelöléseket, és a Shapley-érték formális definícióját illetve alaptulajdonságait tartalmazza, ebben semmilyen új eredmény nem jelenik meg. A második fejezet a dolgozat szempontjából fontos bonyolultságelméleti fogalmakat foglalja össze, a harmadik pedig a Shapley-értéket mint bonyolultságelméleti problémát tárgyalja. Ebben, a már korábban ismert eredmények mellett, néhány új is lesz. A negyedik fejezet a Shapley-érték Monte-Carlo szimulációval való becslésének lehetséges módszereivel foglalkozik, melyet maga Shapley vetett fel eredetileg, és azóta speciális játékosztályokra születtek bíztató eredmények. Az ötödik fejezet a Shapley-érték egzakt kiszámításának lehetőségeivel foglalkozik. A fejezet legfőbb eredménye a Mann és Shapley (1962) által szavazási játékokra javasolt algoritmussal megegyező komplexitású, illetve annál bizonyos esetekben gyorsabb algoritmus bemutatása bővebb játékosztályokra. Az a sejtésünk, hogy ez az algoritmus érdemben már nem javítható.

1. fejezet

Kooperatív játékok és a Shapley-érték

A kooperatív játékok ma megszokott matematikai formalizmusának alapjai Neumann János és Oskar Morgenstern *Theory of Games and Economic Behaviour* című, először 1944-ben megjelent, könyvéből erednek. A Shapley-érték mint a játék általános megoldása 1953 óta ismert (Shapley, 1953). A továbbiakban ezeket ismertetjük röviden. (Részletes és frissebb, modernebb leírásért ld. pl. Peleg és Sudhölter, 2007).

1.1. TU-játékok

Pénzügyi területeken általában olyan speciális konfliktushelyzetekkel nézünk szembe, ahol (egyedi szubjektív hasznosságok helyett), minden pénzben mérhető, mely mindenki számára ugyanannyit ér. Ennek formális leírása a következő.

1.1.1. Definíció. *Átruházható hasznosságú (transferable utility), vagy röviden TU-játékon* (N, v) *rendezett párt értünk, ahol* N *egy nemüres, véges halmaz,* v *pedig* $v : 2^N \to \mathbb{Z}$ *halmazfüggvény, melyre* $v(\emptyset) = 0$*, ahol* 2^N *az* N *halmaz hatványhalmaza, azaz összes részhalmazainak halmaza.*

Az N halmaz elemeit játékosoknak, részhalmazait koalícióknak, a v függvényt pedig karakterisztikus függvénynek nevezzük, a $v(S)$ számot pedig az $S \subseteq N$ koalíció értékének[3]. A játékosok számát, azaz N elemszámát a továbbiakban n-el fogjuk jelölni. A TU jelző, az előrebocsátott megjegyzésnek megfelelően, arra utal, hogy egyedi szubjektív preferenciák helyett,

[3]Mint azt a bevezetőben lévő példáknál láttuk, a karakterisztikus függvény költséget is leírhat, de általánosságban a „koalíció értéke" kifejezést fogjuk használni.

egyetlen, a játékot leíró karakterisztikus függvény létezését tételezzük fel, mely minden lehetséges koalícióhoz hozzárendeli azt a minden játékos számára közös (pénzben mért) értéket, amit az adott koalíció önmagában, a többi játékos nélkül, elő tud állítani.

Legyen N egy nemüres halmaz. Jelölje $\mathcal{G}_N$, az N-hez, mint játékosok halmazához tartozó összes lehetséges TU-játékok (vagy röviden, az N fölötti TU-játékok) halmazát. Más szóval, lássuk el játékosok egy N halmazát az összes lehetséges karakterisztikus függvénnyel, és ezek összességét jelölje $\mathcal{G}_N$. $\mathcal{G}_N$ elemein természetes módon értelmezhető az összeadás művelet: $g_1 = (N, v_1)$ és $g_2 = (N, v_2)$ esetén legyen $g_1 + g_2 = (N, v_3)$, ahol v_3 a v_1 és v_2 pontonkénti (azaz koalíciónkénti) összege, azaz $\forall S \subseteq N$-re $v_3(S) = v_1(S) + v_2(S)$. Hasonlóan értelmezhető a játékokon az egész számokkal vett szorzás, ha $g \in \mathcal{G}_N$ és $c \in \mathbb{Z}$, akkor $cg \in \mathcal{G}_N$ az a játék melynek karakterisztikus függvénye g karakterisztikus függvényének pontonkénti (koalíciónkénti) c-szerese. Könnyen látható, továbbá, hogy pozitív c estén cg ugyanaz, mint $g + g + \cdots + g$ (összesen c példányban). Ezt röviden úgy mondjuk hogy $\mathcal{G}_N$ egy $\mathbb{Z}$-modulus[4].

1.1.2. Definíció. *Legyen $N \neq \emptyset$ halmaz. Érték alatt $\varphi : \mathcal{G}_N \to \mathbb{Q}^N$ függvényt értünk.*[5]

Az *érték* (*value*) kifejezés történelmileg alakult ki megkülönböztetésül a *megoldás* (*solution*) kifejezéstől, mely utóbbi alatt halmaz értékű függvényt értünk. Shapley eredeti cikkében még csak a Shapley-értéket nevezte „érték"-nek (mivel értelemszerűen Shapley-értéknek nem nevezhette), ma ezt a kifejezést általánosabban használjuk. A játék értéke a játék egy lehetséges megoldási koncepcióját jelenti, abban az értelemben, hogy minden játékban, minden játékoshoz hozzárendel egy számot, melyet úgy is lehet interpretálni, hogy az adott játékos ennyit kapjon az osztozkodásnál, ha a játék (a karakterisztikus függvény) megszerezhető pénzt ír le, illetve ennyit fizessen, ha költséget ír le.

Ha $\varphi : \mathcal{G}_N \to \mathbb{Q}^N$ egy érték, akkor definíció szerint minden $g \in \mathcal{G}_N$-re $\varphi(g)$ egy $N \to \mathbb{Q}$ függvény, melybe be lehet helyettesíteni egy $i \in N$ játékost, megkapva így az i játékos $\varphi(g)(i) \in \mathbb{Q}$ részesedését a g játékban. A jelölések egyszerűsítése érdekében a továbbiakban ezt $\varphi_i(g)$-vel, vagy ha nem okoz félreértést, elhagyva a játékra vonatkozó jelölést, φ_i-vel jelöljük.

A karakterisztikus függvény illetve az érték értékkészletéből kizártuk a nem egész, illetve az irracionális számokat. Erre, a pusztán technikai feltételre, ezen a ponton még nem lenne okvetlenül szükségünk (a definíció valós számokkal is értelmes lenne), de irracionális számokat nem tudunk számítógéppel ábrázolni, ezért számítástudományi értelemben nem kapnánk értelmes kérdéseket. A racionális számokat a karakterisztikus függvény esetében az egyszerűség kedvéért zártuk ki, a közös nevezővel átszorozva „lényegében ugyanazt" a játékot kapnánk.

[4] A modulus a vektortérhez hasonló struktúra, csak nem test, hanem gyűrű fölött értelmezzük. Részletesebb leírásért ld. Kiss (2007).

[5] A^B a $B \to A$ függvények halmazát jelöli tetszőleges A, B halmazok esetén, tehát $\mathbb{Q}^N$ olyan függvények halmaza, melyek minden játékoshoz hozzárendelnek egy racionális számot.

A 1.1.2. definíció szerint bármely $\varphi : \mathcal{G}_N \to \mathbb{Q}^N$ függvényt értéknek nevezünk, a játék „értelmes" megoldásának azonban csak bizonyos tulajdonságoknak eleget tevő függvényeket érdemes nevezni. A legfontosabb ilyen tulajdonságokat formalizálja a következő

1.1.3. Definíció. *Legyen $N \neq \emptyset$, $\mathcal{G} \subseteq \mathcal{G}_N$ egy játékosztály, $\varphi : \mathcal{G}_N \to \mathbb{Q}^N$ egy érték Azt mondjuk, hogy φ a $\mathcal{G}$ játékosztályon*

- *megvalósítható, ha minden $g = (N, v) \in \mathcal{G}$-re $\sum \varphi_i(g) \leq v(N)$,*
- *hatékony, ha minden $g = (N, v) \in \mathcal{G}$-re $\sum \varphi_i = v(N)$,*
- *individuálisan racionális, ha minden $g = (N, v) \in \mathcal{G}$-re $\forall i \in N$-re $\varphi_i \geq v(\{i\})$,*
- *koalíciósan racionális, ha minden $g = (N, v) \in \mathcal{G}$-re $\forall S \subseteq N$-re $\sum\limits_{i \in S} \varphi_i \geq v(S)$.*

1.2. A Shapley-érték

Az 1.1.3. definíció a megoldás stabilitását, megvalósíthatóságát leíró tulajdonságokat fogalmazta meg. Shapley eredeti kiindulópontja valamilyen „igazságos" megoldás keresése volt, melynek alapgondolatai, hogy

- ha két játékos egy játék szempontjából „egyforma", akkor abban a játékban ugyanazt a kifizetést kapják,
- ha egy játékos egy játékban „haszontalan", akkor a kifizetése (abban a játékban) nulla legyen,
- ha egy játékot két másik összegére bontunk, és külön-külön játszuk le őket, ezzel senki se járhasson jobban (vagy rosszabbul).

A következőkben ezeket formalizáljuk.

1.2.1. Definíció. *Legyen a $g \in \mathcal{G}_N$ játékban $S \subseteq N$ egy koalíció, $i \in N$ egy játékos, melyre $i \notin S$. Az i játékos S koalíció értékéhez való határ-hozzájárulása (marginal contribution) alatt a*

$$v_i'(S) = v(S \cup \{i\}) - v(S)$$

számot értjük.

Két játékost akkor tekinthetünk „egyformának" egy játékban, ha a határ-hozzájárulásuk megegyezik minden olyan koalícióra nézve, melynek egyik sem tagja. Shapley 1953-as cikkében olyan általános φ megoldást keresett és talált, amely eleget tesz az 1.1.3. definíció szerinti hatékonysági és a fenti három igazságossági feltételnek, melyeket összefoglal az alábbi

1.2.2. Definíció. *Legyen $N \neq \emptyset$, $\mathcal{G} \subseteq \mathcal{G}_N$. Azt mondjuk, hogy a $\varphi : \mathcal{G}_N \to \mathbb{Q}^N$ érték a $\mathcal{G}$ játékosztályon*

- *additív (ADD), ha $\varphi(g_1 + g_2) = \varphi(g_1) + \varphi(g_2)$ minden olyan $g_1, g_2 \in \mathcal{G}$-re, melyre $g_1 + g_2 \in \mathcal{G}$,*
- *egyenlően kezelő (ETP - Equal Treatment Property), ha minden $g = (N, v) \in \mathcal{G}$ játékban, bármely két i és j játékosra, ha $v_i'(S) = v_j'(S)\ \forall\ i, j \notin S \subseteq N$-re, akkor $\varphi_i(g) = \varphi_j(g)$,*
- *hatékony (EFF - Efficiency), ha minden $g = (N, v) \in \mathcal{G}$ játékban $\sum_i \varphi_i(g) = v(N)$,*
- *nulla játékos tulajdonságú (NPP - Null Player Property): ha minden $g = (N, v) \in \mathcal{G}$ játékban, ha egy $i \in N$ játékosra $v_i'(S) = 0\ \ \forall i \notin S \subseteq N$-re, akkor $\varphi_i = 0$.*

Vegyük észre, hogy az ETP, EFF, és NPP tulajdonság játékonként értendő, azaz egy adott rögzített $g \in \mathcal{G}$ játék esetén a $\varphi(g) : N \to \mathbb{Q}$ függvényre szab ki feltételeket, míg az ADD tulajdonság egy függvényazonosság, a $\varphi(g_1 + g_2) = \varphi(g_1) + \varphi(g_2)$ egyenlőség azt jelenti, hogy $\forall i \in N$ játékosra $\varphi_i(g_1 + g_2) = \varphi_i(g_1) + \varphi_i(g_2)$.

A TU-játékok legismertebb megoldási koncepciójának legfőbb eredménye a következő

1.2.1. Tétel (Shapley, 1953). *Minden $N \neq \emptyset$ esetén $\exists!\ \varphi : \mathcal{G}_N \to \mathbb{Q}^N$ érték, mely a teljes $\mathcal{G}_N$-en eleget tesz az ADD, ETP, NPP és EFF tulajdonságoknak.*

1.2.1. Unicitás

Könnyű látni, hogy, az 1.2.1. tétel *ETP*, *NPP*, *EFF* és *ADD* feltételeinek eleget tevő φ függvény legfeljebb egy van (vagyis, ha létezik, akkor egyértelmű). Vezessük be ehhez a következő játékosztályt.

1.2.3. Definíció (egyetértési játékok). *Legyen $\emptyset \neq R \subseteq N$ egy nemüres koalíció, $c \in \mathbb{Z}$ paraméter, és tekintsük azt a $g_{R,c} = (N, v_{R,c}) \in \mathcal{G}_N$ játékot, melyre*

$$v_{R,c}(S) = \begin{cases} c & \text{ha } R \subseteq S, \\ 0 & \text{egyébként.} \end{cases}$$

Az ilyen alakban adott játékokat egyetértési játékoknak (unanimity games) nevezzük.

Megjegyzés. Az egyetértési játékok karakterisztikus függvénye c-ben homogén, azaz

$$c_2 v_{R,c_1}(S) = c_1 c_2 v_{R,1}(S) = v_{R,c_1 c_2}(S) = c_1 v_{R,c_2}(S)$$

minden $S \subseteq N$ koalícióra, speciálisan $g_{R,c} = c \cdot g_{R,1}$.

Első lépésként belátjuk, hogy φ az egyetértési játékok osztályán egyértelműen meghatározott.

1.2.1. Lemma. *Legyen $\emptyset \neq R \subseteq N$, $c \in \mathbb{Z}$, $g = g_{R,c}$ egyetértési játék, és φ olyan érték, melyre $\varphi(g)$ ETP, EFF és NPP. Ekkor minden $i \in N$-re*

$$\varphi_i(g) = \begin{cases} \frac{c}{|R|} & \text{ha } i \in R, \\ 0 & \text{egyébként.} \end{cases}$$

Bizonyítás. Az NPP tulajdonság miatt, ha $i \notin R$, akkor $\varphi_i(g) = 0$. Az ETP tulajdonság miatt minden $i \in R$-re $\varphi_i(g)$ az i-től függetlenül ugyanaz a szám, és az EFF tulajdonság miatt ezek összege $|R|\varphi_i(g) = v_{R,c}(N) = c$. □

Megjegyzés. A φ függvény c-ben homogén, azaz $\varphi(g_{R,c}) = c \cdot \varphi(g_{R,1})$, ami nem túl meglepő, mert ennek pozitív c-re az additivitás miatt egyébként is igaznak kell lennie.

Innen már sejthető a bizonyítás második lépése. Mivel additiv függvény csak egyféleképpen terjedhet ki azon játékosztályra, melynek elemei előállnak egyetértési játékok összegeként, meg kell mutatni, hogy minden játék ilyen. Most ez következik.

1.2.2. Lemma. *A $g_{R,1}$ alakú „normált” egyetértési játékok bázist alkotnak $\mathcal{G}_N$-ben, azaz minden $g \in \mathcal{G}_N$ játék előáll éspedig egyértelműen*

$$g = \sum_{\substack{R \subseteq N \\ R \neq \emptyset}} \lambda_R \cdot g_{R,1}$$

alakban valamely $\lambda_R \in \mathbb{Z}$ együtthatókkal.

Megjegyzés. A lemmában szereplő λ_R együtthatókat Harsányi osztaléknak (Harsányi dividend) nevezzük (Harsanyi 1959, 1963).

Megjegyzés. Ha az 1.1.1. definícióban a karakterisztikus függvény értékkészletéről nem követeltük volna meg, hogy csak egész számok lehetnek, hanem megengedtük volna a racionális (vagy akár valós) számokat, akkor $\mathcal{G}_N$ egy $\mathbb{Q}$ (vagy $\mathbb{R}$) fölötti (2^n-1)-dimenziós vektortér lenne, ami megegyezik az egyetértési játékok számával, így azok – mivel könnyen láthatóan lineárisan függetlenek – bázist alkotnának, amiből egyszerűen következne a kiterjesztés egyértelműsége. Mivel azonban kikötöttük, hogy a karakterisztikus függvények értékei egész számok, a bizonyítás kicsit körülményesebb, viszont így azt is be tudjuk látni, hogy a λ_R együtthatók is egész számok.

Bizonyítás. Tekintsük azt a $2^n \times (2^n-1)$-es M mátrixot, melynek sorai az $S \subseteq N$ összes lehetséges koalíciókkal, oszlopai pedig az összes $g_{R,1}$ ($\emptyset \neq R \subseteq N$) játékokkal vannak indexelve és

melyre $M[S, g_{R,1}] = v_{R,1}(S)$. Tehát az M mátrix oszlopai a $g_{R,1}$ játékok $v_{R,1}$ karakterisztikus függvényei. Azt kell belátni, hogy tetszőleges v karakterisztikus függvény, azaz olyan oszlopvektor, melynek az üres halmazhoz tartozó koordinátája nulla, egyértelműen előáll az M mátrix oszlopainak egész együtthatós lineáris kombinációjaként. Az üres halmazhoz tartozó csupa nulla sor tehát redundáns, ezt elhagyhatjuk. A lineáris kombináció tagjainak (oszlopok) és az egyenletek (sorok) sorrendje nyilván tetszőleges, ezért készítsük el az M mátrixot úgy, hogy az oszlopindexei, a $g_{R,1}$ játékok az R koalíciók elemszáma szerinti növekvő sorrendben legyenek, a sorokban lévő S koalíciók pedig pontosan ugyanebben a sorrendben (az üres halmazhoz tartozó sort már elhagytuk). Mivel $v_{R,1}(S) = 0$, ha $R \not\subseteq S$, egy véges halmaz pedig nem lehet részhalmaza egy nála kisebb elemszámú halmaznak, megegyező elemszámúnak pedig csak akkor, ha egyenlőek, így az M mátrix alsó háromszög mátrix lesz, a főátlóban csupa 1-esekkel (mivel minden halmaz részhalmaza saját magának). Ebből következik, hogy a mátrix determinánsa (a főátlóbeli elemek szorzata) 1 (sorok vagy oszlopok más sorrendje esetén ennek csak az előjele változna). Az M mátrix tehát invertálható és az inverze is egész számokból áll, amiből már nyilván következik a lemma állítása. □

A 1.2.2. lemmából már triviálisan következik φ egyértelműsége, hiszen az additivitás miatt minden $g \in \mathcal{G}_N$ esetén $\varphi(g)$ csak

$$\varphi(g) = \varphi\left(\sum_R \lambda_R \cdot g_{R,1}\right) = \sum_R \varphi\left(\lambda_R \cdot g_{R,1}\right) = \sum_R \varphi(g_{R,\lambda_R})$$

lehet, ez pedig a λ_R együtthatók egyértelműsége miatt értelmes kiterjesztés a teljes $\mathcal{G}_N$-re. Ebből azonban még nem nyilvánvaló, hogy a teljes $\mathcal{G}_N$-re kiterjesztett függvény is eleget tesz az ETP, EFF és NPP feltételeknek, egyelőre csak azt láttuk be, hogy a megadott tulajdonságokkal legfeljebb egy φ érték létezhet.

1.2.2. Létezés

Vegyük észre, hogy a 1.2.1. tétel feltételei közül a hatékonyság az egyetlen, amelyet nem elégít ki az azonosan nulla függvény, meglepő módon tehát, ez tűnik a legkevésbé triviálisnak. Van azonban egy nagyon egyszerű és kézenfekvő módja, hogy ezt elérjük. Tegyük fel, hogy a játékosok egy előre megbeszélt helyen és időben találkoznak, hogy végrehajtsanak egy bizonyos feladatot, majd, mikor a feladat készen van, valahogy elosszák egymás közt az eredmény értékét. Az elsőnek érkező i_1 játékos önmagában $v(\{i_1\})$ értéket tud előállítani, ha a többiek el sem jönnek, ez lesz az ő φ_{i_1} kifizetése. Amikor a második megérkezik, ketten együtt $v(\{i_1, i_2\})$ értéket tudnak elérni, amin a legtriviálisabbnak tűnik $\varphi_{i_1} = v(\{i_1\})$ és $\varphi_{i_2} = v(\{i_1, i_2\}) - v(\{i_1\})$ felosztásban osztozkodni. Ahogy sorban érkezik a többi játékos, ugyanezt folytatják, a j-ediknek érkező játékos részesedése $\varphi_{i_j} = v(\{i_1, i_2, \ldots, i_j\}) - v(\{i_1, i_2, \ldots, i_{j-1}\})$ épp az ő határ-hozzájárulása

az érkezésekor meglévő koalícióhoz, vagyis pont annyit kap, amennyivel annak a közösségnek az értékét növeli, amelyhez csatlakozik. Ennek az „osztozkodásnak" a végeredménye hatékony, mert az elemeinek összege egy teleszkópikus összeg, melynek megmaradó tagjai az első és az utolsó: $v(N) - v(\emptyset) = v(N)$, azonban teljesen esetleges, mert attól függ, milyen sorrendben érkeznek a játékosok, így nem tesz eleget az *ETP* axiómának. A megoldás az, hogy az így kapott (véletlen) kifizetés-vektorokat kiátlagoljuk a játékosok összes lehetséges sorrendje szerint. Ez a nevezetes Shapley-érték.

1.2.4. Definíció. *Legyen $g = (N, v)$ egy TU-játék $n = |N|$ játékossal. A játékosok egy érkezési sorrendjén, vagy röviden sorrendjén vagy permutációján egy $o : N \leftrightarrow \{1, 2, \ldots, n\}$ kölcsönösen egyértelmű függvényt értünk. Ha $i \in N$ egy játékos, o egy sorrend és $o(i) = j \in \mathbb{N}$ ez azt jelenti, hogy az i játékos j-ediknek érkezik. A játékosok összes lehetséges sorrendjeiből álló $\{o : N \leftrightarrow \{1, 2, \ldots, n\}$ kölcsönösen egyértelmű függvény$\}$ halmazt $\mathcal{O}(N)$-el jelöljük. Nyilván $|\mathcal{O}| = n!$. Ha $o \in \mathcal{O}(N)$ egy sorrend, $i \in N$ egy játékos, akkor $S_o(i) \subseteq N$ jelöli az i előtt érkező játékosokat: $S_o(i) = \{i' \in N \mid o(i') < o(i)\}$.*

1.2.5. Definíció. *A $g = (N, v)$ TU-játék Shapley-értéke*

$$Sh_i = \sum_{o \in \mathcal{O}(N)} \frac{v_i'(S_o(i))}{n!} \qquad \forall i \in N\text{-re.} \tag{1.1}$$

A Shapley-érték tehát minden játékban, minden játékoshoz hozzárendel egy racionális számot, tehát az 1.1.2. definíció értelmében *érték*, és könnyű belátni, hogy kielégíti 1.2.1. tétel mind a négy feltételét. (az *ETP* az egyetlen, ami nem teljesen triviális, de az is könnyen meggondolható). A felfedezése[6] óta eltelt közel hetven évben a Shapley-értéknek számtalan egyéb axiomatizálása látott napvilágot, általánosságban, és speciális játékosztályokra egyaránt (bővebben ld. pl. Young (1985), Chun (1989), de Clippel (2018), Dubey (1982) vagy Pintér (2015)). Ezzel a kérdéssel a továbbiakban nem foglalkozunk.

Az i játékos Shapley-értékének kiszámolásakor figyelembevesszük a játékosok összes lehetséges sorrendjét. A $v_i'(S_o(i))$ érték azonban nem változik attól, hogy az i játékost megelőző és követő játékosok egymás közti sorrendje megváltozik, de senki, aki korábban előtte volt, nem kerül mögé és vice versa azaz, ha az i játékos ugyanahhoz a koalícióhoz csatlakozik. Ha ezeket, a nagyon sokszor előforduló, ekvivalens tagokat, melyek száma $|S_o(i)|! \cdot (n - |S_o(i)| - 1)!$ összevonjuk, akkor az

$$Sh_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! \cdot (n - |S| - 1)!}{n!} \cdot v_i'(S) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \frac{1}{\binom{n-1}{|S|}} \cdot v_i'(S) \tag{1.2}$$

formulához jutunk. Shapley eredeti cikkében ez a nehezebben értelmezhető képlet szerepel.

[6] Egyesek szerint „feltalálása", de nem kívánunk 2800 éves matematikafilozófiai vitákat eldönteni ebben a dolgozatban.

2. fejezet

Bonyolultságelméleti háttér

Mielőtt rátérnénk a Shapley-érték kiszámításával kapcsolatos problémák részletes tárgyalására, összefoglaljuk a bonyolultságelmélet alapfogalmait és összefüggéseit. Nem célunk az elmélet precíz felépítése, csak az, hogy egy félig naiv bevezetést adjunk a témában, megkönnyítve a további fejezetek értelmezését mélyebb számítástudományi háttér nélkül is. Részletesebb és precízebb felépítéshez ld. pl. Papadimitriou (1994), illetve Arora és Barak (2009).

2.1. Számítási modell

Az algoritmus leggyakrabban használt matematikai modellje a Turing-gép, most ezt ismertetjük.

- Legyen Σ egy legalább három elemű, véges halmaz, melyet ABC-nek nevezünk, elemeit pedig szimbólumoknak. Feltesszük, hogy az ABC tartalmaz egy *-al jelölt „üres" szimbólumot. Legyen továbbá $\Sigma_0 = \Sigma \setminus \{*\}$. Σ_0 elemeiből képzett véges sorozatokat (beleértve az üres sorozatot is) szavaknak nevezzük, ezek halmazát $\Sigma_0^* = \bigcup_{i=0}^{\infty} \Sigma_0^i$-el jelöljük. Σ_0^* részhalmazait, azaz szavak halmazát nyelveknek nevezzük.

- A Turing-gép *belső állapot*ainak halmaza egy M véges halmaz, mely tartalmaz egy speciális `START` és egy ettől különböző `STOP` állapotot.

- A Turing-gép ezen kívül véges sok, mindkét irányban végtelen *szalag*ból áll, melyek végtelen sok *memóriacellá*ból állnak. A cellák az összes egész számokkal vannak indexelve, azaz minden cella *címe* egy egész (negatív is lehet) szám és minden egész számhoz pontosan egy memóriacella tartozik mindegyik szalagon. Minden cella pontosan 1 darab (Σ-beli) szimbólum tárolására képes.

- Minden szalagon van egy író- és olvasó *fej*, mely egyszerre egyetlen cella tartalmát „látja" és képes átírni, ha akarja. (A fej nem tudja, hogy melyik cellán áll, csak a cella tartalmát

látja, a címét nem.)

- A Turing-gép működését (az általa futtatott programot) az *átmeneti függvényei* határozzák meg. Minden egyes szalaghoz adott egy

$$\beta_i : M \times \Sigma^k \to \Sigma$$

és egy

$$\gamma_i : M \times \Sigma^k \to \{-1, 0, 1\}$$

függvény (vagy egyszerűbben megfogalmazva két $|M| \times |\Sigma|$-as mátrix), ahol k a szalagok száma, valamint egy darab

$$\alpha : M \times \Sigma^k \to M$$

függvény. Feltesszük, hogy $\alpha(\texttt{STOP}, s) = \texttt{STOP}$, $\beta_i(\texttt{STOP}, s) = s$ és $\gamma_i(\texttt{STOP}, s) = 0$ minden $s \in \Sigma^k$ estén. Jelölje $\beta : M \times \Sigma^k \to \Sigma^k$ és $\gamma : M \times \Sigma^k \to \{-1, 0, 1\}^k$ a β_i és γ_i átmenetfüggvényekből, mint koordinátákból álló vektor értékű függvényeket.

A Turing-gép működése során mindig van valamilyen $m \in M$ állapotban és a fejek látnak valamely $s \in \Sigma^k$ szimbólum k-ast, melynek hatására

- minden fej leírja az ő átmenetfüggvénye szerinti $\beta_i(m, s) \in \Sigma$ szimbólumot, és a $\gamma_i(m, s) \in \{-1, 0, 1\}$ értéknek megfelelően egy cellát lép jobbra vagy balra, vagy a helyén marad,
- a gép az $\alpha(m, s) \in M$ állapotba kerül.

A Turing-gép formálisan végtelen ideig működik, de a `STOP` állapotra tett feltevések értelmében, ha egyszer ebbe az állapotba kerül, a továbbiakban már „nem csinál semmit”. A számítás menete a következő:

- Kezdetben a gép a `START` állapotban van, minden fej a saját szalagjának 0 indexű celláján áll és minden cella tartalma a $* \in \Sigma$ üres szimbólum.
- Megadjuk az inputot, mely egy Σ_0 fölötti szó. A gép első szalagjának 0 indexű cellájától balra elhelyezkedő, véges sok cellából álló összefüggő tartományba írunk Σ_0-beli szimbólumokat, vagyis a szalag „elejére” írunk egy szót szimbólumonként. Az input nem tartalmazhatja az üres szimbólumot, különben a gép arra sem lenne képes, hogy megtalálja az input végét. Maga az input természetesen lehet az üres szó.
- Ezután a Turing-gép az idők végezetéig az átmeneti függvényeinek megfelelően működik, azaz minden egyes lépésben átírja (akár saját magára, azaz tulajdonképpen nem írja át) a fejek által látott cellák tartalmát, minden fej lép valamerre (esetleg ott marad, ahol van), és állapotot vált (esetleg ugyanabban az állapotban marad, amiben volt).

A Turing-gép egy működését leírhatjuk a teljes konfigurációinak sorozatával a következő alakban

$$(m_0, x_0, s_0) \to (m_1, x_1, s_1) \to \cdots \to (m_i, x_i, s_i) \to \ldots,$$

ahol $m_i \in M$, $x_i \in \mathbb{Z}^k$ a fejek helyzete és $s_i \in \Sigma^k$, amit a fejek látnak a cellákban, ahol állnak. Egy ilyen sorozat akkor írja le a Turing-gép egy szabályos működését, ha $m_0 = \texttt{START}$, $m_i = \alpha(m_{i-1}, s_{i-1})$ az i-edik képés után az x_i cellák tartalma $\beta(m_i, s_i)$-re változik és $x_i = x_{i-1} + \gamma(m_{i-1}, s_{s-1})$. Az mondjuk, hogy a Turing-gép leáll, ha valaha a `STOP` állapotba kerül, azaz a fenti sorozatban létezik i, hogy $m_i = \texttt{STOP}$ (és innentől a sorozat konstans).

Látható, hogy a Turing-gép maga is leírható egy véges ABC elemeiből képzett véges sorozattal (csak meg kell adni az ABC-jét, az állapotainak halmazát és az átmenetmátrixait, például oszlopfolytonosan), így egy Turing-gép formális leírása is értelmezhető egy Turing-gép inputjaként. Ennek részletes tárgyalásába nem bocsátkozunk, mert rengeteg technikai nehézséggel jár (el kell nevezni Σ-beli szavakkal a gép állapotait, megengedni, hogy egynél több szalag tartalmazzon input-ot, esetleg több szalagot megengedni, vagy kigondolni, hogyan lehet egy szalagon több szalag munkáját szimulálni, stb.), de belátható, hogy

- egy Turing-gép képes szubrutinként meghívni egy másik Turing-gépet, ha annak leírását átadjuk, mint extra inputot egy extra szalagon, továbbá
- létezik U univerzális Turing-gép, mely minden Turing-gép működését képes szimulálni, azaz U-nak átadva bármely másik T Turing-gép leírását és egy x inputot, lefuttatja a T programját x-en, pontosan akkor áll le, ha T is leáll és ekkor U utolsó szalagján ugyanaz van, mint T utolsó szalagján.

Mindezt úgy is interpretálhatjuk, hogy a Turing-gép egy konkrét program (vagy algoritmus) matematikai modellje, az univerzális Turing-gép pedig a számítógép modellje, melyre bármilyen programot lehet írni.

2.2. Algoritmusok bonyolultsága

Jelölje $\Sigma_0^* = \bigcup_{i=0}^{\infty} \Sigma_0^i$ a Turing-gép összes lehetséges inputjainak halmazát, és legyen minden $x \in \Sigma_0^*$-ra $|x|$ az x szó hossza. Ha a T Turing-gép leáll az x inputon, akkor a T állapotainak sorozata az x inputon futtatva

$$\texttt{START} = m_0 \to m_1 \to m_2 \to \cdots \to m_i \to \ldots$$

előbb-utóbb stabilizálódik a `STOP` állapotban. Ebből kifolyólag

- egyrészt, $\exists!\ i = i_x$ csak x-től függő index, melyre $\forall j \geq i$-re $m_j = \texttt{STOP}$, de $m_{i-1} \neq \texttt{STOP}$,

- másrészt, T szalagjainak tartalma nem változik az i-edik lépés után,
  - ▷ speciálisan T utolsó szalagjának (0-tól kezdődő) első L darab cellájának tartalma is ugyanaz az $y_0, y_1, \ldots, y_{L-1}$ szimbólumsorozat marad, ahol L az első *-ot tartalmazó cella címe,
  - ▷ továbbá, ha S_j jelöli a nemüres cellák számát a j-edik lépés után, akkor ezen S_j sem változik az i-edik lépés után, így ezen számoknak van egy S_x csak az input-tól függő maximuma.

A fenti legkisebb $i = i_x$ index a Turing-gép *lépésszáma* vagy *futási ideje* az x inputon, melyet `TIME`(T,x)-el jelölünk, az $y_0, y_1, \ldots, y_{L-1} \in \Sigma_0$ szimbólumokból álló szó a T outputja az x inputon, melyre a függvényekhez hasonló $T(x)$ jelölést használjuk ($L = 0$ esetén ez az üres szó), míg a nemüres cellák maximális S_x száma a futás során a T által felhasznált tárterület nagysága, vagy más szóval T *memóriaigénye*, melyet pedig `SPACE`(T,x)-el jelölünk.

A tár és idő, mint a számítási problémák két szűkös erőforrása nem Turing-gép specifikus, a legtöbb számítási modellben hasonlóan mérjük az algoritmusok hatékonyságát. A formális definíciók modellfüggők a Turing-gép esetében nagyon egyszerűek, más modellekben bonyolultabbak lehetnek, de a tár- és idő bonyolultság koncepciója egységes: egy algoritmus futási ideje a megtett elemi lépések száma (csak az a kérdés, hogyan definiáljuk az „elemi lépés" fogalmát), a memóriaigénye pedig a felhasznált tárterület nagysága, beleértve az inputot is. Hogy ezeket mérni tudjuk, szem előtt kell tartani néhány dolgot.

1. Egy algoritmus futási ideje (és a felhasznált tárterület is) nagymértékben függ(het) az inputtól, de nem arra vagyunk kíváncsiak, hogy milyen gyorsan tudunk egy konkrét számsorozatot sorba rendezni, hanem hogy általánosságban hány lépés n darab számot sorba rendezni.

2. A tár és idő hajszálpontos mérése általában teljesen reménytelen (és különböző modellekben számszerűen eltérhet).

3. A futási idő az input méretével tipikusan nő, ezért nem az az érdekes, hogy „kicsi" inputon mennyi, hanem, hogy *milyen gyorsan* nő az input méretével.

Mindezen okokból egy algoritmus tár- és idő komplexitását úgy definiáljuk, mint a tárhely és a futási idő végtelenbeli (aszimptotikus) viselkedését, az input méretének függvényeként, az adott méretű inputok közül a lehető legrosszabb esetet véve figyelembe. Ehhez szükséges bevezetni még néhány jelölést.

2.2.1. Definíció. *Legyen $f,\ g : \mathbb{N} \to \mathbb{N}$ pozitív értékű függvények. Az mondjuk, hogy $f = O(g)$ (kimondva: „f egyenlő ordó g"[7]), ha $\exists c \in \mathbb{R}$ pozitív konstans és $n_0 \in \mathbb{N}$ küszöbindex, hogy*

[7] Valószínűleg szerencsésebb lenne az $f \in O(g)$, azaz „f eleme ordó g" szóhasználat, de így terjedt el.

$f(n) \leq c \cdot g(n)$ $\forall n \geq n_0$ *esetén. Megfordítva, azt mondjuk, hogy* $f = \Omega(g)$*, ha* $g = O(f)$ *és* $f = \Theta(g)$*, ha* $f = O(g)$ *és* $g = O(f)$ *egyszerre.*

Szemléletesen megfogalmazva, $f = O(g)$ azt jelenti, hogy f a végtelenben nem tart nagyságrendileg gyorsabban végtelenbe, mint g. Könnyű belátni, hogy az $f = O(g)$ reláció tranzitív, továbbá $f = \Theta(g) \Leftrightarrow g = \Theta(f)$ és, ha $f = \Theta(g)$ akkor $f = O(h) \Leftrightarrow g = O(h)$ $\forall h : \mathbb{N} \to \mathbb{N}$ pozitív függvényre. Röviden összefoglalva $f = \Theta(g)$ ekvivalenciareláció, és az $f = O(g)$ reláció részbenrendezés az ekvivalenciaosztályokon. A jelölés azt próbálja megfogalmazni, hogy két függvényt „egyformának" tekintünk, ha „ugyanolyan gyorsan" tartanak végtelenbe, míg, ha az egyik gyorsabban, akkor azt „nagyobbnak" tekintjük.

Legyen most T egy Turing-gép. A korábban bevezetett jelöléssel legyen `TIME(T,x)` a T futási ideje egy x inputon és legyen `SPACE(T,x)` a felhasznált tárterület. Aggregáljuk ezeket az input mérete szerint, azaz legyen

- `TIME(T, n)` $= \max_{|x|=n}$ `TIME(T,x)` a T futási ideje a legrosszabb n méretű inputon,
- `SPACE(T, n)` $= \max_{|x|=n}$ `SPACE(T,x)`, a T maximális memóriaigénye egy n méretű inputon.

A T tár- és idő bonyolultsága a T memóriaigénye a `SPACE(T, n)` illetve a `TIME(T, n)` függvény Θ reláció szerinti ekvivalenciaosztálya. Azaz részletesebben, azt mondjuk, hogy

- T futási ideje (vagy kissé magyartalanul idő komplexitása) $f(n)$ (ahol f egy $\mathbb{N} \to \mathbb{N}$ függvény), ha `TIME(T, n)` $= \Theta(f)$,
- T memóriaigénye (vagy tár komplexitása) $g(n)$ (ahol g egy $\mathbb{N} \to \mathbb{N}$ függvény), ha `SPACE(T, n)` $= \Theta(g)$.

Gyakoribb az a szóhasználat, amikor a tár- és/vagy idő komplexitásra csak felső becslést adunk. Azt mondjuk, hogy

- T $f(n)$ időben fut, ha futási ideje `TIME(T, n)` $= O(f)$,
- T $g(n)$ tárban fut, ha, memóriaigénye `SPACE(T, n)` $= O(g)$.

Megjegyzés. Egy Turing-gép (illetve egy algoritmus valamely más számítási modellben vagy akár egy számítógépes program a gyakorlatban) elvileg akkor is használhat véges mennyiségű memóriát, ha sohasem áll le (végtelen ciklusba kerül). Ezt az érdektelen esetet a definícióban az egyszerűség kedvéért zártuk ki.

Elérkeztünk a legfontosabb definícióhoz.

2.2.2. Definíció. *Azt mondjuk, hogy a* T *Turing-gép futási ideje polinomiális (vagy hogy polinom-időben fut), ha az idő komplexitása* n^k *valamely* $k \in \mathbb{N}$ *pozitív egész számra. Hasonlóan*

T polinom tárban fut, ha a memóriaigénye n^j valamely $j \in \mathbb{N}$ pozitív egész számra.[8] *Konvenció szerint, – ha külön nem specifikáljuk, hogy melyik komplexitásra gondolunk – a „polinomiális" jelző mindig az időre vonatkozik, ha polinomiális tárról beszélünk, ezt külön kiemeljük.*

A polinom-időben futó algoritmusok azok, amelyek az elmúlt évtizedek tapasztalatai szerint a „gyakorlatban működnek", az ennél lassabbak futási ideje tipikusan legalább exponenciális, így nagyon „kicsi" input méretet leszámítva használhatatlanok. Néhány tipikus komplexitási osztály például

- $O(n)$, azaz lineáris: például minimum/maximum keresés,
- $O(\log(n))$: keresés rendezett tömbben (bináris keresés),
- $O(n^2)$, négyzetes (vagy kvadratikus): ilyen néhány naiv rendezési eljárás, mint a beillesztéses rendezés és a buborék rendezés,
- $O(n\log(n))$: optimális rendezési algoritmusok, például összefésüléses rendezés,
- $O(n^3)$ azaz harmadfokú: például Gauss-elimináció.

2.3. Számítási problémák, NP és co-NP osztály

Az eddigiekben vázoltuk, hogyan definiálhatjuk egy algoritmus futási idejét és memóriaigényét (idő- és tár komplexitását), ami az elmélet könnyebb része. Ha ugyanis már van egy konkrét algoritmusunk, annak a komplexitását általában nem túl nehéz felmérni, vagy legalábbis felső becslést adni rá. Azonban egy adott problémáról eldönteni, hogy (a) algoritmikusan megoldható-e egyáltalán, és (b) ha igen, akkor mi ennek a legjobb módja, az nagyon nehéz. (Itt jegyezzük meg, hogy már az sem triviális, hogy mit nevezzünk a „legjobb" algoritmusnak, mert a futási idő és a megmóriaigény néha csak egymás rovására javítható.)

2.3.1. Eldöntési problémák

A továbbiakban legyen Σ egy olyan ABC, mely a * üres szimbólum mellett tartalmazza a 0 és 1 szimbólumokat, melyeket számjegyekként interpretálunk, így fel tudjuk írni tetszőleges pozitív egész szám kettes számrendszerbeli alaját. Legyen a korábbi jelöléseknek megfelelően $\Sigma_0 = \Sigma \setminus \{*\}$ és $\Sigma_0^* = \bigcup_{i=0}^{\infty} \Sigma_0^i$ a Σ_0 fölötti összes lehetséges véges szimbólumsorozatok (azaz szavak) halmaza. Mivel a szavak alkotják egy Σ feletti Turing-gép összes lehetséges inputját és outputját, egy algoritmikus problémát formálisan Σ_0^*-on értelmezett függvényként írhatunk le. Ezeknek két fajtáját különböztetjük meg.

[8] A két kitevőt máshogy jelöltük, mert általában nem egyeznek meg.

- Számítási problémán a következőt értjük. legyen $f : \Sigma_0^* \to \Sigma_0^*$ függvény. Azt kérdezzük, letezik-e T Σ feletti Turing-gép, amely ezt kiszámolja, és ha igen, milyen tár- illetve idő komplexitással. (Azon, hogy egy Turing-gép kiszámol egy függvényt, egyszerűen azt értjük, hogy minden $x \in \Sigma_0^*$ inputra leáll és az outputja éppen $T(x) = f(x)$.)
- Eldöntési problémán a következőt értjük. Legyen $L \subseteq \Sigma_0^*$ egy nyelv, létezik-e olyan Turing-gép mely *eldönti* L-t, azaz kiszámolja az indikátorfüggvényét (azaz az output-ja 1, minden L-beli inputon, egyébként 0), és ha igen milyen komplexitással?

A továbbiakban eldöntési problémákra fogunk koncentrálni. Az mondjuk, hogy az L nyelv eldönthető, ha van olyan Turing-gép, ami eldönti. A számítástudománynak azt a részterületét, amely nyelvek eldönthetőségét illetve Σ_0^*-on értelmezett függvények kiszámolhatóságát vizsgálja, kiszámíthatóság-elméletnek nevezik és a továbbiakban ezzel nem foglalkozunk, mert az itt vizsgált, Shapley-értékkel kapcsolatos problémák ebből a szempontból triviálisak. Ehelyett a következő kérdést vizsgáljuk.

- Legyen $L \subseteq \Sigma_0^*$ egy nyelv, létezik-e olyan Turing-gép mely L-t polinom-időben eldönti, és
- ha igen, pontosan milyen a komplexitása?

Ez a kérdés a számítástudomány bonyolultságelmélet részterületéhez tartozik. Amennyiben a fenti kérdés első felére a válasz igen, akkor azt mondjuk, hogy az L nyelv P-ben van, vagy P-beli. Pongyolán fogalmazva, $P = \{\text{polinom-időben eldönthető problémák}\}$. Kicsit precízebben:

$$P = \{L \subseteq \Sigma_0^* : \ \exists\ T \text{ Turing-gép, mely } L\text{-t Polinom-időben eldönti}\}.$$

A kérdést idő helyett tár komplexitásra is fel lehetne tenni, ezzel szintén azért nem foglalkozunk, mert a Shapley-érték polinom tárban még a definíciója alapján is kiszámolható, úgyhogy az egyetlen releváns kérdés, hogy milyen gyorsan lehet kiszámolni.

2.3.1. Példa. A probléma, amit meg szeretnénk oldani, egy szám prímtényezőkre bontása. Álljon a $\Sigma = \{0, 1, |, *\}$ ABC a bináris számjegyekből, az üres szimbólumból és egy semleges elválasztó karakterből, amit most egy függőleges vonal jelöl. Tekintsük az

$$L = \left\{ n|k \ : \begin{array}{l} n \text{ és } k \text{ egy egész szám kettes számrendszerbeli alakja és} \\ n\text{-nek van 1-től különböző, } k\text{-nál nem nagyobb osztója} \end{array} \right\}$$

nyelvet. Ha ez a nyelv P-ben van, akkor azt a gépet, amely polinom-időben eldönti szubrutinként használva, „felezgetéssel" bármely n számnak megkereshető valamelyik l osztója, majd ugyanúgy n/l valamelyik osztója, amíg van, stb. végül n-t fel tudjuk bontani prímtényezőkre. A tudomány jelenlegi állása szerint nem tudjuk, hogy ez a nyelv P-ben van-e. A nyilvánvalónak tűnő megoldás, hogy kezdjük el 2-től k-ig elosztani n-t minden számmal, nem polinomiális, mert legalább k lépést igényel, míg az input mérete $\lceil \log_2(n) \rceil + \lceil \log_2(k) \rceil + 1$ (a bináris számjegyek száma plusz az elválasztó karakter).

2.3.2. Nemdeterminisztikusan polinomiális problémák

Legyen most L egy nyelv, amit szeretnénk eldönteni, és legjobb barátunk, Háry János, azt állítja, hogy ő minden $x \in \Sigma_0^*$ szóról – jó pénzért – meg tudja mondani, hogy L-ben van-e vagy sem, de azt nem hajlandó elárulni, honnan tudja. Az üzlet létrejöttének semmi akadálya nem lenne, de – bármennyire szép dolog is a bizalom – azért fel kell tennünk a kérdést, hogy mégis miért higgyük el a választ, amit kapunk. János erre a következőt mondja: „Minden egyes $x \in L$ szóhoz átadok egy másik, $y \in \Sigma_0^*$ szót, melynek hossza $|x|$-ban polinomiális és könnyen ellenőrizhetően bizonyítja, hogy x L-ben van.". Ez nem hangzik rosszul, de figyeljük meg, hogy arra vonatkozóan nem hangzott el ígéret, hogy mi a helyzet, ha $x \notin L$.

Az ilyen nyelvek egy nagyon fontos problémaosztályt alkotnak, melynek definíciója a következő.[9].

2.3.1. Definíció. *Azt mondjuk, hogy az L nyelv nemdeterminisztikusan polinomiális, röviden NP-beli, ha minden $x \in L$-hez létezik $y \in \Sigma_0^*$ melyre $|y| = O\left(|x|^k\right)$, és létezik olyan T Turing-gép, mely polinom-időben eldönti az $\{x, y \in \Sigma_0^* \times \Sigma_0^* : x \in L\}$ nyelvet.*

A definícióban szereplő y szót *tanu*nak is szokták nevezni, mert tanusítja, hogy x L-ben van. Ezzel kapcsolatban két dolgot érdemes kiemelni.

- Az NP definíciójában lévő T Turing-gép az „Igaz-e, hogy y bizonyítja, hogy x L-ben van?" kérdésre ad választ polinom-időben. Ez azt jelenti, hogy ha T az (x, y) inputon leáll és „igen" (azaz 1) választ ad, akkor $x \in L$, de ha „nem" (azaz 0) választ ad, az nem azt jelenti, hogy $x \notin L$, hanem azt, hogy „Ez az y nem bizonyítja hogy x L-ben lenne.". Ebből semmi nem következik arra nézve, hogy x valójában L-ben van-e vagy sem, csak annyi, hogy az y erre nem bizonyíték.

- Nem tettük föl, hogy a tanu kiszámolható. A dolog lényege éppen az, hogy nem tudjuk, hogy hogyan találtuk, de ha már a kezünkben van, az egy könnyen ellenőrizhető bizonyíték x L-beliségére.

Vegyük észre, hogy (mint azt előre is említettük) az NP problémaosztály látszólag nem szimmetrikus. A definíció nem követeli meg annak ellenőrizhetőségét, hogy $x \notin L$. Vezessük be a

$$co\text{-}NP = \{L \subseteq \Sigma_0^* : \Sigma_0^* \setminus L \in \mathrm{NP}\}$$

jelölést, és vizsgáljuk meg ezt az aszimmetriát közelebbről. A kiinduló problémánk, miszerint adott egy L nyelv, szeretnénk eldönteni, teljesen szimmetrikus volt, azaz az eldönthető nyelvek

$$\{L \subseteq \Sigma_0^* : \exists T \text{ Turing-gép ami } L\text{-t eldönti}\}$$

[9] A számítástudomány kurzusokon ez általában nem definíció, hanem tétel, de most az egyszerűség kedvéért ezt az ekvivalens jellemzést használjuk definícióként.

osztálya szimmetrikus. Ehhez csak azt kell látni, hogy ha a T Turing-gép eldönti L-t, akkor a $\Sigma_0^* \setminus L$ nyelvet eldönti az a Turing-gép, amely minden x inputra szubrutinként meghívja T-t majd kiírja $1 - T(x)$-et. A

$$P = \{L \subseteq \Sigma_0^* : \exists T \text{ Turing-gép ami } L\text{-t polinom-időben eldönti}\}$$

osztály, ugyanezen okból, szintén szimmetrikus. Az NP osztály azonban nemcsak látszólag, hanem valóban nem szimmetrikus. Ha egy $L \subseteq \Sigma_0^*$ nyelvet úgy interpretálunk, hogy az bizonyos tulajdonsággal rendelkező Σ_0^*-beli szavakból áll, akkor NP azon tulajdonságok osztálya, melyek meglétére létezik polinom-időben ellenőrizhető bizonyíték, míg co-NP azon tulajdonságok osztálya, melyekre létezik polinom-időben ellenőrizhető cáfolat, és ez egy természetes módon aszimmetrikus jelenség.

2.3.2. Példa. Tegyük fel például, hogy egy számról akarjuk eldönteni, hogy prím-e. Legyen

$$L_p = \{n \in \mathbb{N} : n \text{ prím}\}.$$

Nyilvánvaló, hogy $L_p \in$ co-NP, azaz ez a tulajdonság könnyen cáfolható. Erre a tanu egy valódi osztó, mert az oszthatóság könnyen ellenőrizhető. Nem nyilvánvaló azonban, hogy milyen tömör bizonyítékot tudunk elképzelni arra, ha egy szám prím[10].

2.3.3. Példa. Ha faktorizálni akarunk, légyegében fordítva teszünk fel egy hasonló kérdést: legyen

$$L_f = \{n, k : n\text{-nek van 2 és } k \text{ közötti osztója}\}.$$

Ezt bizonyítani könnyű, $L_f \in$ NP, hiszen egy megfelelő osztó jó tanu, de hogyan igazoljuk egy ilyen osztó hiányát?

2.3.4. Példa. Az

$$L_r = \{x_1, x_2, \ldots, x_n, K \in \mathbb{N} : \text{az } x_i\text{-k közül néhánynak az összege } K\}$$

nyelv a nevezetes részletösszeg-probléma (*SSP, subset sum problem*). Ez NP-ben van, hiszen ha már adott az x_i-k egy részhalmaza, akkor az ellenőrzéshez csak össze kell őket adni. De hogyan bizonyítható, az összes exponenciálisan sok eset végigpróbálgatása nélkül, hogy a probléma nem megoldható?

Visszatérve a kiinduló problémára, az $x \overset{?}{\in} L$ kérdés megoldására vonatkozó ajánlat akkor ér valamit, ha $L \in (\text{NP} \cap \text{co-NP}) \setminus \text{P}$ (vagy legalábbis nem tudjuk, hogy $L \in P$-e). Ez esetben a problémát nem tudjuk megoldani, de a megoldást tudjuk ellenőrizni. De vajon van-e ilyen

[10] Valóban nem nyilvánvaló, de közepesen mély számelméleti ismeretek birtokában belátható, hogy a prím tulajdonság NP-beli (sőt mi több, P-beli de ez már egy komoly, nehéz eredmény).

nyelv egyáltalán? Nyilvánvaló, hogy $P \subseteq NP$, hiszen, ha valamit meg tudunk oldani, az egyúttal bizonyítéka is a megoldásnak (ha van olyan Turing-gép, ami tanu nélkül is eldönti $P \ni L$-t, akkor ugyanez a Turing-gép és az üres szó, mint tanu, megfelel az NP definíciójához is). Mivel a P osztály szimmetrikus, az is világos, hogy $P \subseteq \text{NP} \cap \text{co-NP}$. Talán meglepő, de a tudomány jelenlegi állása szerint sem az $\text{NP} \stackrel{?}{=} \text{co-NP}$, sem a $\text{P} \stackrel{?}{=} \text{NP}$ kérdésre nem tudjuk a választ. Ez pongyolán fogalmazva azt is jelenti, hogy

- egyetlenegy olyan konkrét eldöntési problémát sem ismerünk, ahol az „igen" és a „nem" válasz is könnyen ellenőrizhető egy-egy megfelelő bizonyíték birtokában, de a problémát ne tudnánk a bizonyítékok nélkül is gyorsan eldönteni, vagyis
- nem tudjuk, hogy létezik-e olyan eldöntési probléma, aminek megoldását hatékonyan tudjuk ellenőrizni, de tudjuk, hogy megoldani nem lehet, továbbá
- egyetlenegy olyan konrét eldöntési problémát sem ismerünk, ahol az „igen" válaszra van könnyen ellenőrizhető bizonyíték, de biztosan tudnánk, hogy a „nem" válaszra nincsen vagy fordítva (miközben a legtöbb ilyen probléma esetén az egyik triviális a másik pedig teljesen megfoghatatlannak tűnik).

2.4. Visszavezetés és teljesség

A nyelveket, mint eldöntési problémák formális leírását, eddig csak önmagukban, külön-külön vizsgáltuk, az igazán érdekes kérdések azonban a köztük lévő összefüggések. Vegyük észre például, hogy a korábbi jelöléssel az $L_f = \{n, k : n\text{-nek van 2 és } k \text{ közötti osztója}\}$ faktorizáció probléma az $L_p = \{n \in \mathbb{N} : n \text{ prím}\}$ prímtulajdonság-probléma általánosítása, hiszen minden $m \in \mathbb{N}$ esetén $m \in L_p \Leftrightarrow (m, m) \notin L_f$, sőt könnyen látható, hogy valójában $m \in L_p \Leftrightarrow (m, \lceil \sqrt{m} \rceil) \notin L_f$. A következőkben ezt általánosítjuk.

2.4.1. Definíció. *Legyen L_1 és L_2 két tetszőleges nyelv. Azt mondjuk, hogy L_1 polinomiálisan visszavezethető L_2-re, ha létezik olyan T polinomiális futásidejű Turing-gép, melyre minden $x \in \Sigma_0^*$ esetén $x \in L_1 \Leftrightarrow T(x) \in L_2$.*

Az L_1-nek L_2-re való visszavezetése azt jelenti, hogy az L_1 probléma minden példányához könnyen hozzá tudjuk rendelni az L_2 egy példányát, úgy, hogy „igen" példányhoz „igen" példányt, „nem" példányhoz mindig „nem" példányt rendelünk, de anélkül, hogy tudnánk, hogy az adott példány „igen" vagy „nem" példány-e. Az egyik legegyszerűbb, nem teljesen triviális példa erre az alábbi:

2.4.1. Példa. Legyen

$$L_{col3} = \{3 \text{ színnel kiszínezhető gráfok}\},$$

és

$$L_{col17} = \{17 \text{ színnel kiszínezhető gráfok}\}.$$

Egy gráf színezésén a csúcsainak a színezését értjük úgy, hogy szomszédos csúcsok különböző színűek (irányítatlan, egyszerű gráfokról beszélünk). Egyik probléma sem általánosítása a másiknak, de L_{col3} könnyen visszavezethető L_{col17}-re. Ehhez minden gráfhoz kell konstruálni egy másikat úgy, hogy ha az eredeti három színnel színezhető, akkor az új 17 színnel színezhető legyen és vice versa. Ezt könnyen láthatóan úgy kell csinálni, hogy fogjuk az eredeti gráfot meg egy 14 csúcsú teljes gráfot, és minden lehetséges élet behúzunk köztük. Nyilvánvaló, hogy ekkor az eredeti gráf egy 3-színezése kiterjeszthető az új egy 17-színezésévé, míg az új egy 17-színezése az eredetit három színnel színezi, azaz a kétféle színezés egyszerre létezik vagy nem létezik, de nem tudjuk, hogy melyik.

Ha L_1 visszavezethető L_2-re, ez azt is jelenti, hogy L_1 nem nehezebb, mint L_2, legalábbis abban az értelemben, hogy

- ha L_2 eldönthető, akkor L_1 is az,
- ha L_2 P-ben van, akkor L_1 is, valamint
- ha L_2 NP-ben van, akkor L_1 is.

Egyrészt, ha T_2 az L_2-t eldöntő Turing-gép, és T a visszavezetést kiszámoló Turing-gép, akkor az a T_1 Turing-gép, ami minden x inputra először meghívja T-t, majd $T(x)$-en meghívja T_2-t és visszatér ennek értékével, nyilván eldönti L_1-et, és ha T_2 polinom-időben fut, akkor T_1 is. Másrész, minden x input esetén a $T(x)$ L_2-beliségét igazoló tanu egyszersmind x L_1-beliségét is igazolja, hiszen ez a két feltétel ekvivalens, az ezt leellenőrző Turing-gépnek csak össze kell kombinálnia a visszavezetést és $T(x)$ ellenőrzését.

Hogy a jelölés tükrözze a problémák nehézsége közti összefüggést, ha az L_1 nyelv az L_2 nyelvre polinomiálisan visszavezethető, ezt úgy fogjuk jelölni, hogy $L_1 \leq_P L_2$. Mivel két polinomiális visszavezetés egymásutánja is polinomiális visszavezetés, a $\leq_P$ reláció tranzitív, azaz részbenrendezés az összes Σ_0 ABC feletti nyelveken, melyek minimális elemei a P-beli, azaz „könnyű” nyelvek (azaz, ezek bármely más $\emptyset$-tól és a teljes Σ_0^*-tól különböző nyelvre – speciálisan kölcsönösen egymásra is – visszavezethetőek), a maximális elemei pedig a „nehéz” nyelvek, amit pontosabban megfogalmazva a következő fogalmakhoz jutunk.

2.4.2. Definíció. *Azt mondjuk, hogy az L nyelv*

- *NP-nehéz, ha minden $L' \in NP$ nyelvre $L' \leq_P L$,*
- *NP-teljes, ha NP-nehéz és $L \in NP$.*

Az NP-teljesség a bonyolultságelmélet egyik legfontosabb fogalma. Pongyolán fogalmazva, az algoritmikusan „nehéz” problémák osztályát jelenti, de ezzel az interpretációval nagyon vigyázni kell. Egyrészt, vannak problémák, amik eleve nincsenek benne NP-ben, így ezek még nehezebbek. Másrészt, nem tudjuk, hogy P $\stackrel{?}{=}$ NP-e, így az NP-beli problémákról nem állíthatjuk biztosan, hogy nehezek, csak azt, hogy a lehető legnehezebbek. Mivel P $\subseteq$ NP, így NP úgy is interpretálható, mint a problémák legbővebb olyan osztálya, amelyekre elvi lehetőség van polinom-idejű megoldást találni. Az, hogy egy probléma NP-teljes, azt jelenti, hogy egyrészt NP-ben van, azaz, a megoldására létezik gyorsan ellenőrizhető bizonyíték, másrészt, olyan nehéz, hogy ha egyetlen egy ilyet „véletlenül” meg tudnánk oldani polinom-időben, akkor azonnal meg tudnánk oldani a világ összes többi olyan problémáját is, amelyre ez egyáltalán elvileg lehetséges. Vagyis, ha egyetlen NP-teljes problámára lenne polinom-idejű algoritmus, akkor egy csapásra lenne az összes többi NP-beli problémára is (a nem NP-beliekre pedig úgy is lehetetlen). Az, hogy egy probléma NP-nehéz, nagyjából ugyanezt jelenti, kivéve, hogy még azt sem tudjuk róla, hogy NP-ben van-e. Ezekre tipikus példák az NP-teljes problémák komplementerei.

A definícióból nem nyilvánvaló, hogy léteznek-e egyáltalán NP-teljes nyelvek, illetve, ha igen, akkor csak a létezésüket tudjuk-e bizonyítani, vagy meg is tudunk adni ilyet konrétan. Nyilvánvaló, hogy ha L_1 NP-teljes, és $L_1 \leq_P L_2$, akkor L_2 is az (feltéve, hogy NP-beli), így ha már van egyetlenegy NP-teljes nyelvünk, akkor ennek polinomiális visszavezetésével igazolhatjuk sok másik nyelv NP-teljességét is. De hogyan találjuk meg az elsőt?

Cook tétele

Tekintsük ismét az L_{col3} 3-színezhetőség problémát és egy k csúcsú $G(V,E)$ gráfot. Jelölje k a csúcsok számát, vezessük be az $r_1, g_1, b_1, r_2, g_2, b_2, \ldots, r_k, g_k, b_k$ $3k$ darab változót, és tekinsük a

$$\bigwedge_j (r_j \vee g_j \vee b_j) \wedge \bigwedge_j (\bar{r}_j \vee \bar{g}_j) \wedge (\bar{r}_j \vee \bar{b}_j) \wedge (\bar{g}_j \vee \bar{b}_j) \wedge \bigwedge_{(j,l)\in E} (\bar{r}_j \vee \bar{r}_l) \wedge (\bar{g}_j \vee \bar{g}_l) \wedge (\bar{b}_j \vee \bar{b}_l)$$

logikai kifejezést (ahol a felülvonás a logikai tagadást jelöli). Könnyen látható, hogy ez elkódolja a gráf 3-színezhetőségét: minden j-re a j-edik csúcs legyen piros, ha r_j igaz, zöld, ha g_j igaz és kék, ha b_j igaz. Az r_j, g_j és b_j változók logikai értékei pontosan akkor választhatóak meg úgy, hogy a formula igaz legyen, ha ez egy jól definiált és helyes színezés, mert minden j-re r_j, g_j és b_j közül pontosn az egyik igaz, és ha két csúcs között van él, akkor ez mindig két különböző színt jelölő változó lesz.

A logikai formulák kifejezőereje valójában sokkal nagyobb. Nem csak a 3-színezhetőséget, hanem tetszőleges Turing-gép átmenetfüggvényei által meghatározott működési szabályokat képesek elkódolni. Ez a nevezetes Cook-tétel: az úgynevezett `SAT` nyelv NP-teljes (Cook, 1971).

Ez volt az első és talán egyetlen, melynek NP-teljességét közvetlenül a Turing-gépek felhasználásával lehetett (és kellett) bizonyítani. Ahogy ez megvolt, és felismerték a jelentőségét, további nyelvek NP-teljességét már a korábbiakra visszavezetve garmadával lehetett bizonyítani (Karp, 1972), és ma már hemzsegnek az NP-teljes nyelvek (Garey és Johnson, 1979). Ezek közül a legnevezetesebbek például a korábban már említett részletösszeg probléma és a gráfok három (illetve minden legalább három) színnel való színezhetősége, a Hamilton-kör probléma, a hátizsák-probléma vagy az egészértékű programozás.

2.5. Pszeudopolinomiális futási idő

Azt mondjuk, hogy a T Turning-gép futási ideje pszeudopolinomiális, ha az inputjában lévő számokat egyes számrendszerben megadva a futási ideje polinomiális. Ez nagyjából azt akarja jelenteni, hogy T futási ideje az input értékének, de nem feltétlenül a méretének polinomja. A különbség akkor a legszembetűnőbb, ha az input egyetlen n szám, ami tetszőleges $K \geq 2$ alapú számrendszerben megadható $O(\log_K(n)$ darab szimbólummal, így, ha a Turing-gép éppen n lépést tesz, akkor a futási ideje pszeudopolinomiális („pszeudolineáris", $O(n)$), de ez az input méretének, ami $O(\log(n))$ exponenciális függvénye.

Hogy az ilyen algoritmusokat még véletlenül se nézzük polinomiálisnak, annak legnagyobb jelentősége a kriptográfiában van[11]. Tekintsük például azt a naiv algoritmust, ami úgy próbál egy n számot faktorizálni (prímtényezőkre bontani), hogy 2-től $n-1$-ig egyesével végigellenőriz minden számot, hogy osztója-e n-nek[12]. Magának az oszthatóságnak az ellenőrzését elhanyagolva (nem ez a szűk keresztmetszet ennél a problémánál), marad a „számoljunk el egyesével n-ig" algoritmus. 100-ig a legtöbben egy-két perc alatt el tudunk számolni. 200-ig számolni nagyjából kétszer annyi idő. Csakhogy a komplexitást elvileg nem úgy kell mérni, hogy megduplázzuk az inputot, hanem hogy megduplázzuk az input hosszát. A 100 egy háromjegyű szám, ha az inputot leíró szimbólumok számát akarjuk megduplázni, akkor (minimum) ahhoz a problémához jutunk, hogy számoljunk egy 100000-ig. A dolgozat megírásának időpontjáig (2023 tavasz) mintegy 27 milliós megtekintettsége van (bár valószínűtlen, hogy bárki is tényleg végignézte), annak a 2017-ben készült YouTube videónak, amelyen egy „Harvard" felíratú kék pólót viselő srác elszámol százezerig.[13] A kiírás szerint az eredeti videó több, mint negyven órás. Ha a „számoljunk el n-ig" algoritmus lineáris lenne, akkor nem 200-ig, hanem 100000-ig tartana kétszer annyi ideig elszámolni, mint 100-ig. Az algoritmus exponenciális, de pszeudopolinomiális, mert a futási ideje az input numerikus értékének polinomiális függvénye.

[11] A Shapley-érték kapcsán is fontos lesz, de itt most jobb egy egyszerűbb példa

[12] Eltekintünk attól, hogy valójában elég $\sqrt{n}$-ig keresni az osztókat.

[13] Bár nincs rá garancia, hogy örökre fent marad, jelenleg még megtekinthető itt: https://www.youtube.com/watch?v=xWcldHxHFpo

Hogy mire jó a pszeudopolinomiális algoritmus, ha valójában exponenciális futásidejű, azt célszerű egy példával illusztrálni. Tekintsük a korábban már említett részletösszeg problémát: adott $n+1$ darab $x_1, x_2, \ldots, x_n \in \mathbb{N}$ és $K \in \mathbb{N}$ pozitív egész szám, és azt kérdezzük, ki lehet-e választani az x_i-k közül valamennyit, hogy az összegük éppen K legyen. Ez a probléma nyilván NP-beli, hiszen ha valaki megmondja mely számokat kell kiválasztani, akkor elég könnyű ellenőrizni, hogy az összegük valóban egyenlő-e K-val, de mint korábban említettük NP-teljes. A kézenfekvő megoldás (*nyers erő*vel[14]) végigpróbálgatni az x_i-k 2^n darab lehetséges részhalmazát, ami legalább exponenciális futásidejű (nem csak n-től függ, ha az x_i-k nagyok, az összeadás komplexitását sem hanyagolhatjuk el). Van azonban egy trükk. Mivel az x_i-k mind nemnegatív számok, minden részhalmazuk összege 0 és W közé esik, ahol $W = \sum x_i$ az összesnek az összege. Hozzunk létre egy $W+1$ elemű $[A_0 = 1, A_1 = 0, \ldots, A_W = 0]$ tömböt, és töltsük ki a következőképpen: $\forall\, i \in \{1, 2, \ldots, n\}$-re $\forall\, j \in \{x_i, x_{i+1}, \ldots, W\}$-re ha $A_{j-x_i} = 1$, akkor legyen $A[j] = 1$. Tehát először csak $A_0 = 1$, majd $i = 1$-re a belső j-s ciklus 1-re változtatja A_{x_1}-et, $i = 2$-re a belső ciklus 1-re változtatja A_{x_2}-t és $A_{x_1+x_2}$-t, és így tovább. Ezzel az ún. dinamikus programozás módszerrel, az i-edik iteráció után azon A_ℓ értékek változnak 1-re melyek előállnak az első i szám valamely részhalmazának összegeként. A részletösszeg-probléma tehát akkor és csak akkor megoldható, ha az utolsó iteráció után $A_K = 1$. A megoldáshoz használt A tömböt összesen n-szer írtuk felül, minden felülírás (a belső j-s ciklus) legfeljebb W művelet W-nél kisebb számokon, így a teljes futási idő $O(nW\log(W))$, ahol tehát $W = \sum x_i$. Ez kissé unortodox módja a komplexitás megadásának, mert azt definíció szerint az input méretének függvényében kell megadni, ami pedig $\sum \log(x_i)$ lenne, az n pedig ennek nem függvénye. Ezzel együtt, könnyen látható, hogy a futási idő nem becsülhető felülről az input méretének semmilyen polinomjával, az algoritmus „nagyjából exponenciális". Az érdekes benne az, hogy a brute force végigpróbálgatás $\Omega(2^n)$ futási idejéhez képest ez n-ben „kicsi", a gond a W szorzó, mely az input méretében exponenciálisan nagy és n-től függetlenül akármilyen nagy is lehet. Ez elég jól rávilágít, hogy milyen komplexitási tulajdonságot próbál megfogni a pszeudopolinomiális idő komplexitás: olyan problémák összesége, ahol az input valamilyen számokból álló sorozat, de nem azért nehéz megoldani, mert túl sokan vannak, hanem azért mert túl nagyok. A részletösszeg és sok „hasonló" probléma, beleértve a Shapley-érték bizonyos speciális eseteit, melyek inputjai egész számokból álló sorozatok, nem a paraméterek számának növekedésével nehezednek (nagyon), hanem elsősorban attól nehezek, hogy gigantikus paramétereket adunk meg inputnak (sok kis számra könnyebb megcsinálni, mint kevés nagyra), ami a gyakorlati alkalmazásokban nem feltétlenül áll fenn.

[14] Az ilyen „összes eset végigpróbálgatása" módszert *brute force* algoritmusnak nevezik.

2.6. Közelíthetőség

Eldöntési problémák közelítésének semmi értelme nincsen, mert mit jelentene az, hogy valami „körülbelül igaz”? A közelítést általában optimalizálási problémák esetében szokták értelmezni, de itt, most egy sokkal egyszerűbb, speciális esetet fogunk érteni rajta.

Legyen f egy $f : \mathbb{Z}^* \to \mathbb{Q}$ függvény, ahol $\mathbb{Z}^* = \bigcup_{i=0}^{\infty} \mathbb{Z}^i$, azaz f minden egész számokból álló véges sorozathoz hozzárendel egy racionális számot. Ha valamely $\varepsilon > 0$ valós számhoz létezik olyan T_ε Turing-gép, amely polinom-időben fut és minden x inputra $|f(x) - T_\varepsilon(x)| \leq \varepsilon |f(x)|$ (ahol $|\ |$ most a racionális számok abszolútértékét jelöli), akkor azt mondjuk, hogy f közelíthető legfeljebb ε relatív hibával.

Noha léteznek „nagyon jól” közelíthető „nehéz” problémák (olyanok, melyek eldöntési verziója[15] is NP-teljes), ebből nem érdemes elhamarkodott következtetéseket levonni. Egyrészt, a közelítés nem felcserélhető a visszavezetéssel, másrészt, az hogy egy probléma közelíthető, nem azt jelenti, hogy van egy polinomiális algoritmus, ami adott ε-ra kiszámol egy közelítést, hanem azt, hogy adott ε-ra létezik egy polinomiális algoritmus, ami kiszámol egy közelítést. Tehát lehetséges, hogy (1) létezik egy közelítés $\varepsilon = \frac{1}{2}$-re, de $\varepsilon = \frac{1}{3}$ már nem tudjuk hogy van-e, vagy tudjuk, hogy nincs, illetve (2) elképzelhető, hogy létezik egy közelítés $\varepsilon = \frac{1}{2}$-re is és $\varepsilon = \frac{1}{3}$-ra is, de az már egy teljesen más algoritmus, mondjuk az első lineáris, a másik viszont négyzetes. Még ha igaz is, hogy egy probléma minden $\varepsilon > 0$ esetén közelíthető ε hibával, ez sem azt jelenti, hogy van egy polinom-idejű algoritmus ami akármilyen pontossággal közelíti, hanem azt, hogy minden hibához van egy polinom-idejű algoritmus, ami legfeljebb akkora hibával közelíti, de ezek az algoritmusok különbözőek sőt különböző bonyolultságúak lehetnek. Harmadrészt, még abban az esetben is, ha van egy fix algoritmus, ami minden ε-ra kiszámol egy közelítést, ennek a komplexitása, lehet hogy például $\frac{n^2}{\varepsilon}$ (vagy, esetleg $n^2 \cdot 10^{\frac{1}{\varepsilon^2}}$), ami minden fix ε-ra n-ben polinomiális, azonban, ha a probléma egy fix példányára szeretnénk a hibát nullához közelíteni, az már nem megy. A definíció nem zárja ki, hogy a közelítés futásideje $\varepsilon \to 0$ esetén exponenciálisan gyorsan tartson végtelenbe.

[15] Eldöntési verzió alatt azt érthetjük például, hogy létezik-e $x : f(x) \leq K$ adott K inputra.

3. fejezet

A Shapley-érték mint bonyolultságelméleti probléma

Adott egy TU-játék, számoljuk ki a Shapley-értékét! Ez látszólag nagyon egyszerű, be kell helyettesíteni az (1.1) vagy az (1.2) formulába, és megkapjuk a konkrét számokat. Három játékos és nyolc lehetséges koalíció esetén ezt akár kockás papíron is könnyen megtehetjük, de ha a játékosok száma akár csak 100, és a Shapley-értéket algoritmikusan, számítógéppel szeretnénk kiszámolni, akkor mindkét 2. fejezetben tárgyalt problémával szemben találjuk magunkat:

1. Idő: Az utóbbi években (2016-2020) egy szuperszámítógép számítási kapacitása hozzávetőlegesen 100 petaFLOPS[16], ami másodpercenként kb. 10^{17} lebegőpontos művelet elvégzését jelenti.

- Az (1.1) formula egy $n!$ tagszámú összeg, ez azt jelenti, hogy
  - kb. 10 játékosra ki tudjuk számolni,
  - kb. 33 játékosra lehetett volna kiszámolni az ősrobbanás óta eltelt mintegy 14 milliárd év alatt,
  - 100 játékos esetén pedig szó szerint az idők végezetéig tartana.[17]
- Az (1.2) formulának „csak” 2^{n-1} tagja van, ez $n = 100$ esetén $2^{99} \geq 10^{29}$ művelet, ami kb. 10^{12} másodpercig, azaz így is több mint 30 ezer évig tart.

2. Tár: Egy TU játékot elvileg a karakterisztikus függvénye definiál. Ha a játékot a számítógépnek inputként az összes koalíció értékének felsorolásával akarnánk megadni, akkor ez, az említett 100 játékos esetén, mintegy kétmillió gigabyte-nyi adat lenne.

[16] https://en.wikipedia.org/wiki/FLOPS

[17] Tovább tartana mint mire az univerzum eléri a termodinamikai egyensúly (maximális entrópia vagy *hőhalál*) állapotában a 10^{-30}K hőmérsékletet és megszűnik az idő.

3.1. A Shapley-érték bonyolultsága

Mivel egy algoritmus komplexitása alatt a futása során felhasznált erőforrások (tár és idő) maximális mennyiségének nagyságrendjét, mint az input méretének függvényét értjük, a fejezet elején feltett kérdés alapvetően értelmetlen amíg meg nem mondjuk, hogy mi az input, azaz milyen formában, milyen módon van adva a játék. A teljesség kedvéért kezdjük az alábbi, közismert[18] eredménnyel:

3.1.1. Tétel. *Amennyiben egy TU-játékot az összes koalíció értékének felsorolásával adunk meg, a Shapley-érték kiszámítható polinom-időben és tárban.*

Bizonyítás. A bizonyításhoz vizsgáljuk meg, hogy az (1.2) formula hány művelettel számolható ki (az (1.1) formula kiértékelése könnyen láthatóan nem polinomiális). Az input méretét (azaz a koalíciók értékeit megadó elemek összes bitjeinek számát) jelölje $\vartheta = \sum_{\emptyset \neq S \subseteq N} \log_2(v(S))$, mely tehát legalább 2^n, de lehet sokkal nagyobb is. Nyilvánvaló, hogy 2^{n-1} darab határ-hozzájárulást (azaz, ennyi darab kivonást) lineáris időben ki lehet számolni, és a határ-hozzájárulásokat lineáris tárban eltárolni. Mivel az (1.2) formulában legrosszabb esetben is $n!$ választható az összeadandó törtek közös nevezőjének, $|S|! \cdot (|N| - |S| - 1)!$ pedig ennél nem nagyobb, a feladat 2^{n-1} olyan törtnek az összeadása, melyeknek már van közös nevezőjük, a számlálójuk pedig (összesen $\vartheta \cdot n \log(n)$ idő alatt kiszámolható) legfeljebb $n \log(n) + \vartheta$ bites számok. Két szám összeadásakor az összeg legfeljebb egy helyiértékkel lehet nagyobb a legnagyobb összeadandónál, így több szám összeadásakor a bitek száma legfeljebb az összeadandók számának logaritmusával nőhet, ami jelen esetben n, aminél ezek a számok már eleve nagyobbak, így az összeg is egy legfeljebb $n \log(n) + \vartheta$ bites szám. Ebből kell összeadni 2^{n-1} darabot, ez legfeljebb $(n \log(n) + \vartheta) \cdot 2^n$ művelet. Mivel $\vartheta \geq 2^n$ így ez legfeljebb ϑ^2 művelet, és nyilván legfeljebb ugyanennyi tárhely. □

Megjegyzés. A bizonyítás nagyon nagylelkűen bánik az inputot alkotó számok nagyságával, az utolsó lépésben minden határ-hozzájárulást egy legfeljebb annyi bites számnak tekint, amekkora a teljes input mérete, ezért kaptunk ilyen rossz eredményt, négyzetes futási időt. A becslés azonban nem tűnik a koalíciók számában lineárisra javíthatónak, még abban az esetben sem, ha a karakterisztikus függvény 0 - 1 értékű (az input mérete ilyenkor 2^n). Ezesetben ugyanis a határ-hozzájárulások összesúlyozásakor a műveletek száma nem 2^{n-1}, mert – noha a határ-hozzájárulások konstans méretűek – a súlyokat adó faktoriálisok nagy, $n \cdot \log(n)$ bites számok, ami a lineárisnál egy kicsit rosszabb $n \log(n) \cdot 2^n$-re rontja a teljes futási időt.

Összefoglalva tehát, ha egy TU-játékot a koalíciók értékeinek listájával adunk meg, akkor a Shapley-érték négyzetesnél semmiképpen sem rosszabb futási idővel kiszámolható. A tétel nem

[18] A tétel mondhatni folklór, az irodalomban való első felbukkanását keresni sem érdemes.

oldja meg a Shapley-érték problémáját (tehát érdemes tovább olvasni a dolgozatot), ugyanis ebben a formában csak nagyon „kicsi" játékokat szokás definiálni, a legtöbb TU-játékot nem így adjuk meg. Nézzünk néhány konkrét példát.

3.1.1. Példa.

Szavazási játék: Legyen $n \in \mathbb{N}$, $N = \{1, 2, \ldots, n\}$, $w = (w_1, w_2, \ldots, w_n) \in \mathbb{N}^n$ és $q \in \mathbb{Z}$. Minden $S \subseteq N$-re legyen

$$v(S) = \begin{cases} 1, & \text{ha} \quad \sum\limits_{i \in S} w_i \geq q, \\ 0, & \text{ha} \quad \sum\limits_{i \in S} w_i < q. \end{cases}$$

Repülőtér-játék: Legyen $n \in \mathbb{N}$, N egy n elemű halmaz, minden $i \in N$-re $c_i \in \mathbb{N}$, és legyen a karakterisztikus függvény $v(S) = \max\{c_i | i \in S\}$.

Csődjáték: Let $A \in \mathbb{Z}^+, n \in \mathbb{N}, \quad l_1, l_2, \ldots l_n \in \mathbb{N}$ és legyen (N, v) az a TU-játék, melynek karakterisztikus függvénye $v(S) = \max\{0, A - \sum_{j \notin S} l_j\}$.

A játékok részletes leírása, a paraméterek interpretációja, számtalan egyéb példával együtt, megtalálható például Driessen (1988) könyvében.

Egy TU-játékot tehát a karakterisztikus függvénye definiál, de – amint ezt a fenti három példánál is láthatjuk – ez a függvény általában nem az összes koalíció értékének felsorolásával van adva, hanem valami egyszerű képlettel, melyet a játékosok számában polinomiális (a fenti három példánál lineáris) mennyiségű paraméter határoz meg. Ezt formalizálja az alábbi

3.1.1. Definíció. *Tekintsük TU-játékok egy* $G \subseteq \bigcup\limits_{n \in \mathbb{N}} \mathcal{G}_{\bar{n}}$ *halmazát, ahol* $\forall n \in \mathbb{N}$*-re* $\bar{n} = \{1, 2, 3, \ldots, n\}$*. Azt mondjuk, hogy* G *polinomiálisan reprezentálható, ha létezik* Σ *(véges) ABC*[19] *és egy* Σ *feletti legalább három szalagos* T_G *Turing-gép, valamint* $\forall g = (\bar{n_g}, v_g) \in G$*-re olyan* $x_g \in \Sigma_0^*$ *szó, melynek hossza* n_g*-ben polinomiális, és* T_G *első szalagjára egy* n_g *hosszú 0-1-sorozatot, második szalagjára pedig* x_g*-t írva, polinom-időben kiszámolja (bináris alakban visszaadja output-ként) a* g *játékban annak a koalíciónak az értékét, melynek indikátor-vektora az első szalagján van. Röviden* $T_G(S, x_g) = v_g(S)$*.*

A polinomiális reprezentáció létezése azt jelenti, hogy a játékosztály „tömöríthető", azaz leírható a játékosok számában polinomiális mennyiségű adattal. Az 3.1.1. példa játékai mind ilyenek, ha a játékokat meghatározó w_i, c_i, illetve l_i súlyok a játékosok számában legfeljebb exponenciálisan nagyok, hiszen a játékokat meghatározó egyszerű képletek nyilván kiszámolhatóak Turing-géppel.

[19] Mely a 2. fejezet feltevéseinek megfelelően az $*$ üres szimbólum mellet tartalmazza a 0 és 1 szimbólumokat, hogy számokat tudjunk ábrázolni.

3.1.2. Definíció. *Legyen $G \subseteq \bigcup_{n\in\mathbb{N}} \mathcal{G}_{\bar{n}}$ egy polinomiálisan reprezentálható játékosztály. Ekkor a definícióban szereplő x_g szavak egy $L_G = \{x_g \in \Sigma_0^* \mid g \in G\}$ nyelvet alkotnak. Ezt az L_G nyelvet a definícióban szereplő T_G Turing-géppel együtt, az (L_G, T_G) rendezett párt a G játékosztály polinomiális reprezentációjának nevezzük.*

Megjegyzés. A polinomiális reprezentáció nem feltétlenül egyértelmű.

3.1.3. Definíció. *Azt mondjuk, hogy a $G \subseteq \bigcup_{n\in\mathbb{N}} \mathcal{G}_{\bar{n}}$ játékosztályra a Shapley-érték polinom-időben kiszámolható, ha létezik G-nek (L_G, T_G) polinomiális reprezentációja, melyhez létezik olyan T legalább három szalagos Turing-gép, melynek első szalagjára inputként egy $x_g \in L_G$ szót, a második szalagjára pedig T_G leírását írva, (T_G-t szubrutinként használva), polinom-időben kiszámolja a g játék Shapley-értékét.*

Megjegyzés. A Turing-gépek szalagjainak számára tett feltevés természetesen technikai jellegű, és ha ezt elhagynánk, ekvivalens definíciókhoz jutnánk.

Egy játékosztályra a Shapley-érték kiszámolhatósága két feltétel együttes teljesülését jelenti. Egyrészt, a karakterisztikus függvények megadhatóak valamilyen egyszerű képlettel vagy tömören leírható eljárással (az összes koalíció értékének exponenciálisan hosszú listája nélkül), tehát a problémát fel tudjuk írni „sok" játékos esetén is. Másrészt, ebből a tömör reprezentációból, mint inputból, a játék Shapley-értéke valamilyen algoritmussal hatékonyan kiszámolható. Mivel a polinomiális reprezentáció általában nem egyértelmű, ez felvet bizonyos kérdéseket.

1. Hogyan tudjuk eldönteni, hogy egy polinomiális reprezentáció tényleg „azt a játékot írja le, amit akartunk", azaz, hogyan lehet két polinomiális reprezentációról eldönteni, hogy ugyanazt a játékot reprezentálják-e (vagyis ugyanazt a karakterisztikus függvényt számolják-e ki)?

2. Előfordulhat, hogy egy játékosztály kétféleképpen is reprezentálható, de az egyikből ki lehet számolni a Shapley-értéket, a másikból viszont nem. Ezesetben nyilván egyik reprezentációból nem lehet kiszámolni a másikat.

3. Egyáltalán nem egyszerű annak az eldöntése, hogy melyik a könnyebb feladat: adott reprezentációhoz Shapley-értéket számolni, vagy más reprezentációt keresni.

Megjegyzés. A definícióból természetesen kimaradnak olyan érdektelennek tűnő esetek, ha például a játék nem az összes koalíció értékének felsorolásával, de polinomiálisnál több, mondjuk $2^{\sqrt{n}}$ paraméterrel írható le. Emellett kizártunk olyan érdekesnek tűnő esetek is, mint például a Faigle és Kern (1993) által bevezetett ládapakolásos játékok vagy utazóügynök-játékok. Ezek a játékok megadhatóak polinom-sok paraméterrel, azonban már az egyes koalíciók értékének

kiszámolása is NP-nehéz. Ebből nem következik automatikusan, hogy a Shapley-érték sem számolható ki polinom-időben, hiszen az szimmetrikus esetre például triviális, de a továbbiakban ilyen játékokkal sem foglalkozunk.

3.2. Kiszámolható játékok

Bizonyos játékok Shapley-értéke méretüktől függetlenül könnyen kiszámítható.

3.2.1. Példa. A $g = (N, v) \in \mathcal{G}_N$ játékot additívnak nevezzük, ha bármely két diszjunkt $S_1, S_2 \subseteq N$ koalíció esetén $v(S_1 \cup S_2) = v(S_1) + v(S_2)$.

Könnyen látható, hogy additív játék Shapley-értéke: $\varphi_i = v(\{i\})$, ugyanis minden határhozzájárulás is ezzel egyenlő. Az additív játékok megadhatóak a 3.1.1. definíciónak megfelelő alakban, a $\gamma_i = v(\{i\})$ számok, (a játékosok számával megegyező méretű paraméterhalmaz), egyértelműen definiálják a teljes játékot.

3.2.1. Költségallokáció fákon

Az egyik legkorábbi nemtriviális eredmény a Shapley-érték komplexitásával kapcsolatban Megiddo 1978-as *Computational Complexity of the Game Theory Approach to Cost Allocation for a Tree* című cikkében jelent meg.

Legyen a játékosok halmaza $N = \{1, 2, \ldots, n\}$ és legyen $G(N \cup \{0\}, E, d)$ irányított súlyozott élű gyökeres fa, ahol a 0 csúcs a gyökér, és jelölje $d(i, j) \geq 0$ a súlyozást az éleken. Legyen minden $S \subseteq N$-re $v(S)$ az összes olyan él súlyainak összege, melyeket valamelyik S-beli csúcshoz vezető, gyökérből induló út tartalmaz. A súlyok az éleken jelenthetik például egy csatorna-, vasút- vagy elektromos hálózat építési vagy fenntartási költségeit.

A megoldás a Shapley-érték additivitásán alapul. Tekintsük ugyanis minden $e \in E$ élre azt a G_e játékot, amit úgy kapunk az eredetiből, hogy az e él súlyán kívül mindegyiket lenullázzuk. Mivel a karakterisztikus függvény az élek súlyozására nézve additív, ezzel az eredeti játékot n darab sokkal egyszerűbb játék összegére bontottuk, amelyek Shapley-értékei simán összeadódnak. A kapott G_e játékok egyetértési játékok, az 1.2.1. fejezet jelölésével R azon csúcsok halmaza, melyekhez a gyökérből vezető út az e élen átmegy. Ezen csúcsok között kell egyenlően szétosztani az e él költségét. Az összes lehetséges e élre ezeket összeadva, a megoldás úgy is interpretálható, hogy minden él költségét egyenletesen kell szétosztani azon játékosok között, akik az élet használják. Nyilván minden e élre és i csúcsra kiszámolható, például az e végpontjából indított szélességi vagy mélységi kereséssel annak az indikátora, hogy az i csúcs használja-e az e élet és ez az információ eltárolható például egy $n \times n$-es mátrixban, amiből a játékosok Shapley-értékei már könnyen kiszámolhatóak. Mindez a legnaívabb módon is $O(n^2)$

időben megvalósítható. Valójában, Megiddo (1978) szerint, a feladat az algoritmus ügyes optimalizálásával, egyetlen mélységi kereséssel, lineáris időben is megoldható.

3.2.2. További példák

A bevezetőben ismertetett példák közül a variancia-allokációs játék Shapley-értékének komplexitása szintén polinomiális, sőt a megoldás nagyon szemléletes, minden eszköz Shapley-értéke a hozamának a teljes portfólió (saját magát is beleértve) hozamával vett kovarianciája (Colini-Baldeschi et al., 2018), speciálisan a játék pontosan akkor additív, ha az eszközök hozamai függetlenek, és ekkor minden eszköz Shapley-értéke a saját varianciája.

A már ismert játékok közül érdemes még megemlíteni a 3.1.1. példában leírt repülőtérjátékokat. Ezekre az 5. fejezetben még visszatérünk, mert érdekes összefüggésekre fognak rávilágítani. Most csak annyit jegyezzünk meg, hogy a Shapley-értékük könnyen számolható a következő egyszerű észrevétel felhasználásával: minden $c \in \{0, c_1, c_2, \ldots, c_n\}$ számra, a legfeljebb c értékű koalíciók száma $2^{|\{\leq c \text{ költségű játékosok}\}|}$. Ebből differenciálással megkapható a pontosan c_i költségű koalíciók száma, amiből a Shapley-érték az (1.2) formula felhasználásával kiszámolható.

3.3. NP-nehéz játékok

A Shapley-érték kiszámítása a bonyolultságelméleti szempontból másik végletnek tekinthető NP-nehéz problémák között is képviselteti magát. Ezzel kapcsolatban, régi klasszikus eredmények helyett, most néhány frisset mutatunk be.

3.3.1. Tartozásos játékok

A *tartozásos játékok* a 3.1.1 példa csődjátékához nagyon hasonló játékosztály, melyet Csóka és Herings (2019) vezetett be. A játék karakterisztikus függvénye

$$v(S) = \begin{cases} \min\{A, \sum\limits_{j \in S \setminus \{0\}} l_j\} & \text{ha} \quad 0 \in S, \\ \max\{0, A - \sum\limits_{j \notin S \cup \{0\}} l_j\} & \text{ha} \quad 0 \notin S. \end{cases}$$

ahol $N = \{0, 1, \ldots, n-1\}$, A és $l_1, l_2, \ldots l_{n-1}$ pozitív paraméterek, melyekre $A < \sum l_i$. A játék szereplői egy 0-val jelölt játékos, például egy cég, és véges sok hitelező, l_i jelöli a kötelezettséget az i-edik hitelező felé. Az A paraméter a 0-s játékos eszközeinek értéke, melynek értéke kevesebb, mint a kötelezettségek összege, így a cég csődben van. A karakterisztikus függvény azt írja le, hogy az egyes koalíciók mennyit tudnak behajtani az összes követelésükből. A csődbemenő cég is szereplője a játéknak és azok a koalíciók, melyeknek ő is egyik tagja, elsőbbséget élveznek

a kötelezettségek kielégítésében. A 0 játékost nem tartalmazó koalíciók csak a maradékon osztozkodhatnak. A játékkal kapcsolatos további részletekért és példákért ld. Csóka (2018). Egy 2022-ben megjelent cikkben (Csóka et al., 2022) belátjuk az alábbi tételt:

3.3.1. Tétel. *NP-nehéz annak eldöntése, hogy két tartozásos játékban a csődbemenő cég Shapley-értéke megegyezik-e, azaz az* $\{(\mathcal{L}_1, \mathcal{L}_2) \mid Sh_1 = Sh_2\}$ *nyelv NP-nehéz, ahol* $\mathcal{L}_i$ *tartozásos játékok,* Sh_i *pedig a* 0*-s játékos Shapley-értéke az egyes játékokban.*

Bizonyítás. Felhasználjuk, hogy a

$$\text{SUBSET-SUM} = \left\{ (a_1, a_2, \ldots, a_k, M) \in \mathbb{Z}^{k+1} \mid \exists H \subseteq \{1, 2, \ldots, k\} \mid \sum_{i \in H} a_i = M \right\}$$

nyelv NP-teljes. Könnyű belátni, hogy ennek a

$$\text{HALFSUM} = \left\{ (a_1, a_2, \ldots, a_k) \in \mathbb{Z}^k \mid \exists H \subseteq \{1, 2, \ldots, k\} \mid \sum_{i \in H} a_i = \frac{\sum_{i=1}^{k} a_i}{2} \right\}$$

speciális esete is az. Ezt fogjuk visszavezetni a tartozásos játékok Shapley-értékére.

Legyen $(a_1, a_2, \ldots, a_k) \in \mathbb{Z}^k$. Legyenek a kötelezettségek $l_i = a_i$, $A = \frac{\sum a_i}{2}$ és legyen $\mathcal{L}_1$ egy tartozásos játék az l_i és A paraméterekkel. Az $\mathcal{L}_2$ játékban ugyanezek lesznek a kötelezettségek, de az eszközök értékét 1-el csökkentjük. Azt fogjuk megmutatni, hogy az $\mathcal{L}_1$ és $\mathcal{L}_2$ játékban a 0 játékos Shapley-értéke pontosan akkor különböző, ha $(a_1, a_2, \ldots, a_k) \in$ HALFSUM.

Hitelezők egy $S \subseteq \{1, 2, \ldots, n-1\}$ koalíciójának, hitelezők halmazára vonatkozó komplementerét jelölje $S^c = \{1, 2, \ldots, n-1\} \setminus S$. A könnyebb érthetőség kedvéért nevezzük hitelezők egy S koalícióját erősnek, ha a követeléseik összege meghaladja az eszközök értékét: $\sum_{i \in S} l_i > A$, közepesnek, ha éppen egyenlő: $\sum_{i \in S} l_i = A$ és gyengének, ha kevesebb: $\sum_{i \in S} l_i < A$. Mivel az eszközök értéke éppen fele az összes követelésnek, minden S esetén S és S^c közül vagy pontosan az egyik erős, a másik gyenge, vagy mindkettő közepes. Egy gyenge koalíció értéke önmagában nulla, mert a követelések hátrasoroltak, és a komplementer koalíciót nem lehet kifizetni. Amikor a 0 játékos egy ilyenhez csatlakozik, akkor viszont az összes követelés kifizethető, ezért a 0 játékos határ-hozzájárulása megegyezik a követelések összértékével. Egy erős koalíció összes követelése akkor sem elégíthető ki, miután a 0 játékos csatlakozott a koalícióhoz, ezért a komplementer koalíció teljes követelését, mely ilyenkor kevesebb az eszközök értékénél áthozza onnan, tehát ilyenkor ez lesz a határ-hozzájárulás. Ezt tömören megfogalmazva:

$$v_0'(S) = \begin{cases} \sum_{i \in S} l_i, & \text{ha } S \text{ gyenge,} \\ \sum_{i \in S^c} l_i, & \text{ha } S \text{ erős.} \end{cases}$$

Ha S közepes, akkor a két eset egybeesik és $v_0'(S) = A$ természetesen. Ezt felhasználva a 0 játékos Shapley-értéke az $\mathcal{L}_1$ játékban a nevezőben lévő $n!$-al felszorozva:

$$n! \cdot Sh_0 = \sum_S z_S \cdot v_0'(S) = \sum_{S \text{ erős}} z_S \cdot v_0'(S) + \sum_{S \text{ közepes}} z_S \cdot v_0'(S) + \sum_{S \text{ gyenge}} z_S \cdot v_0'(S) =$$

$$= \sum_{S \text{ erős}} z_S \cdot \left(\sum_{i \in S^c} l_i \right) + \sum_{S \text{ közepes}} z_S \cdot A + \sum_{S \text{ gyenge}} z_S \cdot \left(\sum_{i \in S} l_i \right)$$

Ahol $z_S = |S|! \cdot |S^c|!$ pozitív súlyok. Nézzük, mi változik, ha a Shapley-értéket az $\mathcal{L}_2$ játékban akarjuk kiszámolni. Ebben a kötelezettségek ugyanazok, de 1-el csökkentjük az eszközök A értékét. Egy eredetileg gyenge koalíció továbbra is nullát ér, mert a komplementer követelések kevesebb eszközzel még kevésbé fizethetőek ki. Amikor a 0 játékos a koalícióhoz csatlakozik, továbbra is kifizethető az összes követelés, mert az A-nál szigorúan kevesebb volt, tehát legfeljebb $A-1$. A 0 játékos határ-hozzájárulása tehát, és így az összeg harmadik tagja nem változik. Hasonlóan könnyű meggondolni, hogy ugyanez a helyzet az erős koalíciókkal, tehát az összeg első tagja sem változik. Az összeg második tagja pontosan akkor nemüres, ha az eredeti HALFSUM probléma megoldható. Egy közepes koalíció mindkét játékban nullát ér, mert a komplementer követelések elérik ($\mathcal{L}_2$-ben meghaladják) az eszközök értékét, így nem marad semmi. Amikor a 0 játékos egy ilyen koalícióhoz csatlakozik, annak értéke $\mathcal{L}_1$-ben éppen A lesz, $\mathcal{L}_2$-ben pedig éppen $A-1$. Az összeg második tagjának tehát minden tagja, s így a tejles összeg is csökken. Ha tehát van közepes koalíció, akkor a 0 játékos Shapley-értéke az $\mathcal{L}_2$ játékban kisebb lesz az $\mathcal{L}_1$-hez képest, és pont ezt akartuk belátni. □

Ezek szerint nincs polinom-időben futó egzakt algoritmus a tartozásos játékok Shapley-értékének kiszámolására, hacsak nem `P=NP`.

3.3.2. Szavazási játékok közelíthetetlensége

A 3.1 fejezet első példája talán a legismertebb és legfontosabb játékosztály, a szavazási játékok osztálya. Ebben n játékos mindegyike rendelkezik egy a_n súllyal, valamint adott egy q kvóta. A játékosok eldöntendő kérdésekben szavazhatnak, és ha az igennel szavazók összsúlya eléri a kvótát, akkor az indítványt megszavazták. A karakterisztikus függvény azt írja le, mely koalícióknak van elég ereje egy indítvány megszavazásához, azaz mely koalíciók alkotnak többséget.

A szavazási játékok osztálya a Shapley-érték szempontjából a bonyolultságelméleti értelemben „nehéz" problémák közé tartozik. Deng és Papadimitriou (1994) szerint a Shapley-érték kiszámítása `#P`-teljes[20]. Ehelyett most egy, bizonyos értelemben gyengébb, bizonyos értelemben erősebb állítást fogunk igazolni, melynek nem találtuk nyomát az irodalomban.[21]

[20] Ez az NP-teljes eldöntési problémák „kiszámítási" megfelelője, vagyis „nehéz".

[21] Talán azért, mert nehézségét tekintve inkább feladatgyűjteménybe való mint tudományos publikációba.

3.3.2. Tétel. *Az*

$$\Big\{(\mathcal{L}, i) \mid \mathcal{L} = (N, v) \text{ szavazási játék }, i \in N \text{ egy játékos, melyre } Sh_i \neq 0\Big\}$$

nyelv NP*-teljes.*

Bizonyítás. A bizonyítás hasonló a tartozásos játékokról szóló tételéhez, csak sokkal egyszerűbb. Legyen $(a_1, a_2, \ldots, a_k, M)$ a SUBSET-SUM probléma egy példánya, és rendeljük hozzá az $((a_1, a_2, \ldots, a_k, 1, M+1), k+1)$ problémát, vagyis vegyünk egy új játékost, akinek 1 a súlya és állítsuk $M+1$-re a kvótát. Így az 1 súlyú játékos Shapley-értéke pontosan akkor nem nulla, ha a többi játékosnak van olyan koalíciója, amelyben a súlyok összege éppen M. Másrészt a probléma NP-ben van, mert egy ilyen koalíció megadása bizonyítja a Shapley-érték nemnullaságát. □

Vegyük észre, hogy a szavazási játékok Shapley-értékének nemnullasága a részletösszeg problémánál általánosabb

$$\Big\{(a_1, a_2, \ldots a_k, I, J) \in \mathbb{Z}^{k+2} \mid \exists H \subseteq \{1, 2, \ldots, k\} \mid \sum_{i \in H} a_i \in [I, J]\Big\}$$

problémát oldja meg. A részletösszeg probléma ennek az a speciális esete, ha $I = J$. A tétel tehát tulajdonképpen triviális, és ebben a formában nem is túl izgalmas, az alábbi következménye azonban sokkal érdekesebb.

3.3.1. Következmény. *Semmilyen $\varepsilon < 1$-re nem létezik a szavazási játékok Shapley-értékére polinom-idejű ε-közelítés, hacsak nem* P=NP*.*

Bizonyítás. A közelítés definíciója szerint, ha létezne ilyen $\hat{s}$, melyre $s \cdot (1 - \varepsilon) < \hat{s}$ teljesülne, akkor ez $\varepsilon < 1$ miatt szigorúan pozitív becslést adna minden olyan játékos Shapley-értékére, akire ez nem nulla, és $\hat{s} < s \cdot (1 + \varepsilon)$ miatt nullát azokra, akikre nulla, amivel eldöntene egy NP-teljes problémát. Ez valóban csak P=NP esetén lehetséges. □

Megjegyzés. Abból, hogy egy játékosztály esetén nem tudunk polinom-idejű algoritmust a Shapley-érték kiszámolására, nem következik, hogy az csak a brute force 2^{n-1} futásidejű algoritmussal oldható meg. Például éppen szavazási játékokra Klinz és Woeginger (2005) bemutat egy eljárást, mely a koalíciók ügyes „feldarabolásával" meg tud spórolni egy csomó számolást és még mindig exponenciális, de kicsit gyorsabb, $O\left(n\sqrt{2}^n\right)$ időben képes kiszámolni a Shapley-értéket.

3.3.3. Shapley NP-ben?

Alapítunk a bevezetőben látott EGK-hoz hasonló játékot[22], ezúttal nem hét, hanem hetvenhét ország részvételével. A legkisebb résztvevő ország, Aprajafalva (továbbiakban A) és a legna-

[22] Vannak ma is hasonlók, például az EU bizonyos döntéshozó szervei.

gyobb, Bergengócia (továbbiakban B) képviselői kisvártatva összevesznek, az előbbi ugyanis azt állítja, hogy az ő országának Shapley-értéke nulla, míg az utóbbi azt, hogy ez nem igaz.

A Shapley-értéket, mint tudjuk, nehéz kiszámolni, de elképzelhető, hogy a játékosok együttesen rendelkeznek az ehhez szükséges számítási kapacitással, azonban nem bíznak egymásban annyira, hogy a végeredményt elhiggyék. Ha egy játékra „rábizonyítható" lenne a Shapley-értéke, azaz a Shapley-érték NP-beli lenne, akkor a probléma a bizalomhiány ellenére áthidalható lenne. A résztvevő országok együttes erővel kiszámolják a Shapley-értéket és előállítják az ellenőrzéséhez szükséges bizonyítékot. Az NP definíciója szerint a bizonyíték ellenőrzése polinom-időben elvégezhető, ezt már önállóan minden ország a többiek közreműködése nélkül is meg tudná csinálni, és többé nincs vita.

A tudomány jelenlegi állása szerint teljes bizonyossággal nem állíthatjuk, de nagyon valószínűtlen, hogy a Shapley-érték NP-ben legyen. Visszatérve ugyanis az eredeti problémára, B játékos állítására létezik bizonyíték. Annak igazolására, hogy A játékos Shapley-értéke nem nulla, egyszerűen meg kell adni egy olyan koalíciót, melynek értéke kisebb, mint a kvóta, de A-val kiegészítve, már nagyobb.[23] Megfordítva: ha A játékos Shapley-értéke tényleg nulla, hogyan tudja ezt könnyen bizonyítani? Elég megfoghatatlan a dolog, mert rengeteg koalícióról kéne valahogy belátni, hogy a szavazóereje nem esik bele egy bizonyos intervallumba. A bevezetőben Luxemburg példája mutatta, hogy bizonyos speciális esetekben ez belátható. Ha például A kivételével minden játékos súlya és a kvóta is osztható ugyanazzal az m számmal (például mindegyik páros), az A játékos súlya pedig ennél az m-nél kisebb, akkor A Shapley-értéke nulla, és ez az m szám, erre egy tömör bizonyíték. Azonban ez természetesen csak egy elégséges és nem szükséges feltétel, ha ilyen m szám nincs, milyen könnyen ellenőrizhető bizonyítékot tudunk elképzelni? Valószínű, hogy semmilyet. A 3.3.2. tétel szerint ugyanis ez a probléma NP-nehéz. Ha egyúttal NP-beli is lenne, az eldöntené a nevezetes $\texttt{NP} \stackrel{?}{=} \texttt{co-NP}$ problémát.

3.3.2. Következmény. *Hacsak nem* `NP = co-NP`*, akkor a TU-játékok Shapley-értéke nem* `NP`*-beli probléma.*

3.4. Reprezentációs ekvivalencia

Hetvenhét országból álló szövetségünk tagjai heves vita után arra jutnak, hogy a legkisebb ország, A súlyát meg kéne emelni 3-mal. Ezt A és B is elfogadhatónak tartja, két másik ország képviselői azonban interpellálnak. Narnia (a továbbiakban N) képviselője tiltakozik a döntés ellen, arra hivatkozva, hogy A súlyának a növelése az N Shapley-értékét csökkentené, míg

[23] Általánosabban, egy darab nullától különböző határ-hozzájárulás bizonyítékként szolgál a Shapley-érték nemnullaságára minden olyan játékban, ahol a határ-hozzájárulások nemnegatívak. De milyen bizonyítékot tudunk elképzelni abban az esetben, ha a határ-hozzájárulások tetszőleges előjelűek lehetnek?

Középfölde (a továbbiakban K) képviselője azt állítja, hogy a szakértőik szerint A súlyának növelése valójában egyáltalán nem változtatja meg a játékot, és javasolja egy bizottság felállítását ennek kivizsgálására.

A TU-játékok bonyolultságával kapcsolatos utolsó állításunk (melyhez hasonlónak nem találtuk nyomát a szakirodalomban), hogy a K által javasolt bizottság felállítása indokolt, mert a felvetett probléma (teljes általánosságban) nehéz.

3.4.1. Tétel. *NP-nehéz annak eldöntése, hogy a 3.1.1 példa szerint megadott két szavazási játék reprezentáció ugyanazt a játékok definiálja-e.*

Bizonyítás. Részletesen kiírva, azt kell eldönteni, hogy $2n + 2$ darab $w_1, w_2, \ldots, w_n, q$ és $w'_1, w'_2, \ldots, w'_n, q'$ pozitív egész szám esetén igaz-e hogy minden $S \subseteq \{1, 2, \ldots, n\}$-re

$$\sum_{i \in S} w_i \geq q \Leftrightarrow \sum_{i \in S} w'_i \geq q'.$$

Ez pedig a 3.3.1. és 3.3.2. tételhez hasonlóan a részletösszeg-probléma egy újabb köntösben, ugyanis ez $w_i = w'_i$ és $q' = q - 1$ speciális esetben is azzal ekvivalens, hogy $\nexists S \subseteq \{1, 2, \ldots, n\}$ hogy $\sum_{i \in S} w_i = q$ □

3.5. Összefoglaló

A TU-játékokat tehát nem triviális „könnyen használható" formában megadni, ha sikerül, nem feltétlenül tudjuk megmondani, hogy ezek a reprezentációk mikor definiálják ugyanazt a játékot, a Shapley-érték általában nem számolható ki polinom-időben egzaktul, nem adható rá bizonyíthatóan jó közelítés, és még ha valaki megmondaná is az eredményt, valószínűleg nem tudjuk ellenőrizni. Csupa negatív eredmény. Két kiút igérkezik: a Shapley-érték Monte-Carlo szimulációval való becslése, illetve pszeudopolinomiális algoritmusok alkalmazása. A következő két fejezetben ezekkel foglalkozunk.

4. fejezet

Becslés Monte-Carlo szimulációval

A Shapley-érték definíció szerint a játékosok átlagos határ-hozzájárulása az összes koalíció értékéhez a játékosok minden lehetséges sorrendjét figyelembe véve. Ezt úgy is mondhatjuk, hogy a Shapley-érték a várható határ-hozzájárulás egy egyenletes eloszlásból származó véletlen sorrend esetén, tehát egy diszkrét valószínűségi változó várható értéke. A várható érték pedig – ha kiszámolni nem is tudjuk – megbecsülhető Monte-Carlo szimulációval.

A Monte-Carlo szimuláció lényege, hogy egy valószínűségi változó várható értékét, a nagy számok erős törvényére alapozva, egy véletlen mintából a mintaátlaggal becsüljük. Akkor érdemes használni, ha a várható érték kiszámítására nincs egyszerű módszerünk (például analitikusan nem kezelhető) és akkor tudjuk használni, ha a változó olyan eloszlásból származik, amiből (számítógéppel) könnyen tudunk mintát generálni. A Shapley-érték ennek tökéletes példája olyan játékok esetén, amikor nincs ötletünk az exponenciálisan sok tag összegének kiértékelésére, de a karakterisztikus függvény könnyen kiszámolható (tehát pontosan abban az esetben, amellyel az egész dolgozatban foglalkozunk), hiszen egy véges halmaz elemeiből permutációkat generálni könnyű és lineáris idejű probléma.[24]

A fejezet hátralévő részeiben azt a kérdést vizsgáljuk, hogy teljes általánosságban, illetve speciális TU-játékosztályokra hogyan lehet a Monte-Carlo szimuláció becslési hibáját csökkenteni, illetve arra felső korlátot adni. A fejezet fő eredményeként ismertetjük az Illés és Kerényi (2019) által bemutatott algoritmust, mely a már Shapley által 1960-ben (Mann és Shapley, 1960) szavazási játékokra javasolt módszerek nagyfokú általánosításának (is) tekinthető, és tetszőleges játékra és mindenféle heurisztika nélkül alkalmazható.

[24] A véletlen permutációk, illetve általánosabban, visszatevés nélküli minta generálására szolgáló algoritmus leginkább *Fisher–Yates shuffle* (Fisher–Yates keverés vagy Fisher–Yates algoritmus) néven ismert, a szakirodalomban legelőször még a Turing-gép előtt, 1938-ban bukkan fel (Fisher et al., 1938), majd a mai jelöléseinkhez hasonló, felismerhető formában – feltehetően a korábbiaktól és egymástól is függetlenül – bemutatja Durstenfeld (1964) és E. Knuth (1969) is.

4.1. Monte-Carlo szimuláció egyszerű, független mintavételezéssel

Az egyszerű véletlen mintavétel, az elnevezés alapján könnyen kitalálhatóan, mindössze annyit jelent, hogy választunk egy $m \in \mathbb{N}$ mintaelemszámot, legenerálunk m darab független permutációt a játékosok N halmazán, kiszámoljuk az ezekhez tartozó határ-hozzájárulásokat és ezek átlagát tekintjük a Shapley-érték egy becslésének. Castro et al. (2009) részletesen leírja a módszer nagy számok törvényéből és a centrális határeloszlás-tételből következő tulajdonságait (torzítatlan, konzisztens és asszimptotikusan normális), bár, amint arra Maleki et al. (2013) is rávilágít, a konfidencia-intervallumok megadásánál elhanyagolja a CHT maradéktagját, így az nem teljesen korrekt, csak jó közelítés. Maleki et al. (2013) két korrekt konfidenciaintervallumot is megad a határ-hozzájárulások szórásának, illetve alsó és felső korlátainak függvényében. Ezzel együtt, a fejezet hátralévő részeiben - összhangban a szakirodalommal - nem foglalkozunk a becsült Shapley-érték eloszlásának alakjával, csak a szórását igyekszünk csökkenteni. A független minta szórása természetesen $\sigma/\sqrt{m}$, ahol σ a határ-hozzájárulások szórása.

Két becslési módszer összehasonlítása úgy igazságos, ha az őket előállító algoritmusok valamilyen értelemben ekvivalensek. A Monte-Carlo szimulációhoz feleslegesnek tűnik mind az m darab permutációt egyszerre tárolni (erre egyik módszernél sincs szükség), a szűk keresztmetszet inkább a futási idő, mint a memória. Ezért a továbbiakban két algoritmust akkor tekintünk összehasonlíthatónak, ha a futási idejük várható értéke asszimptotikusan egyenlő, tehát mindként algoritmus által elvégzett műveletek átlagos száma n-ben polinomiális, és a polinomok főegyütthatói megegyeznek. Ha ez a feltétel fennáll, akkor azt a módszert tekintjük jobbnak, amelyik kisebb szórású becslést ad a Shapley-értékre. Az összehasonlítás alapját az egyszerű véletlen mintavételezéses eljárás képezi, melynek komplexitása $O(m \cdot (n + G_n))$, ahol m a generált minta elemszáma (ebben a fejezetben ezt a jelölést fogjuk végig alkalmazni), n szokás szerint a játékosok száma, és G_n a karakterisztikus függvény kiértékelésének bonyolultsága (a játékosok számának függvényében).

4.2. Shapley és Mann heurisztikái szavazási játékokra

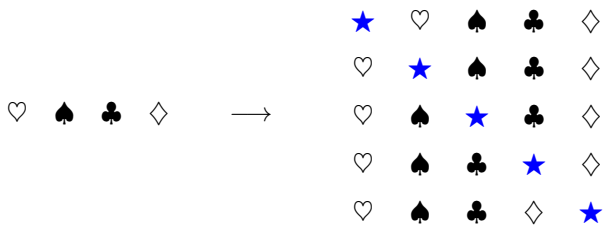
Legyen $\pi_1, \pi_2, \ldots, \pi_m$ független véletlen permutációi (egyenletes eloszlásból) a játékosok N halmazának és $i \in N$ egy tetszőleges játékos. Ha $s_1, s_2, \ldots, s_m$ az i játékos megfelelő határhozzájárulásai valamely TU-játékban, akkor az s_i-k független azonos eloszlású, véges várható értékű és szórású valószínűségi változók, így ezek $\bar{s} = \sum s_i/n$ mintaátlaga a nagy számok törvénye szerint 1 valószínűséggel tart a valódi várható értékhez, azaz az i játékos Sh_i Shapley-értékéhez, míg $\sqrt{m} \cdot (\bar{s} - Sh_i)$ határeloszlása normális, nulla várható értékkel és s_j szórásával.

Ez természeti törvény, melyen semmilyen trükkel nem tudunk változtatni. Hogyan lehetséges akkor jobb (kisebb szórású) becslést adni a Shapley-értékre? Nyilvánvaló, hogy a két feltétel, a függetlenség és az azonos eloszlás közül valamelyiket (esetleg mindkettőt) el kell engedni. Mann és Shapley (1960) a következő szellemes eljárást javasolja:

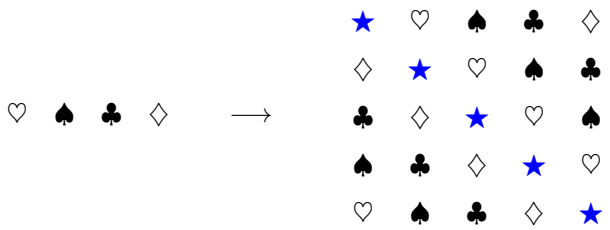
4.2.1. Példa (Shapley-Mann MC algoritmusok)**.**

1. Generáljunk egy véletlen sorrendet az i-től különböző játékosokból.

2. ***A. verzió:*** Tegyük be az i játékost az összes lehetséges helyre a sorban. Ez n darab permutációja az összes játékosnak.



   B. verzió: Tegyük be az i játékost az első helyre, majd az összes játékos így kapott sorrendjét permutáljuk ciklikusan minden lehetséges módon (amíg a ciklus körbe nem ér), vagyis hozzuk előre az utolsó játékost, és küldjünk mindenkit eggyel hátrébb, amíg i az utolsó helyre nem kerül. Ez n darab permutációja az összes játékosnak.



3. Ismételjük ezt meg m/n-szer függetlenül, és az így kapott sorrendekből becsüljük a Shapley-értéket.

Az így generált permutációk (és a belőlük kapott határ-hozzájárulások) nyilván nem függetlenek, ezért a Shapley-értékre kapott becslésünk tulajdonságait többé nem határozza meg a centrális határeloszlás-tétel. Mann és Shapley nem foglalkozik annak formális bizonyításával, hogy a becslés konzisztens és torzítatlan, sőt arról sem állít semmit, hogy jobb-e mint az először szintén általuk javasolt független minta használata. A módszert kizárólag szavazási játékokra használják, és nincs is rá semmi garancia, hogy egyéb játékosztályokra is működik. A teszteléshez használt játék az amerikai elnökválasztás modellje, melynek súlyai a népszámlálási adatok függvényében minden választáskor módosulnak.[25]

[25] A súlyok az első elnökválasztásig, 1788-ig visszamenőleg megtalálhatóak az interneten: `https://en.wikipedia.org/wiki/United_States_Electoral_College`.

4.3. További Variancia-redukciós módszerek

Variancia-redukción a Monte-Carlo szimuláció olyan változatait értjük, melyekkel csökkenteni lehet a mintaátlag szórását a független mintához képest. Ehhez általában szükségünk van valami plusz információra, vagy valamilyen heurisztikán alapuló trükkre. Ilyen heurisztikát talált például Mann és Shapley az előző alfejezetben. Nézzünk erre néhány további példát:

4.3.1. Példa (Függő változó módszere)**.** Legyen Y egy valószínűségi változó $\mu = \mathbb{E}(Y)$ véges várható értékkel, melyet nem ismerünk de Y-ból tudunk mintát generálni. A mintaátlag szórását csökkenthetjük, ha ismerünk Y-ra egy $Y = \beta X + u + \varepsilon$ lineáris regressziós modellt, ahol X az Y-al korrelál, a β regressziós együttható és X várható értéke ismert, továbbá az (X, Y) együttes eloszlásából tudunk mintát generálni. Ekkor Y helyett egyből az Y-al megegyező várható értékű $Y - \beta \cdot (X - \mathbb{E}(X))$ maradéktag várható értékét becsüljük meg, mert ennek várható értéke megegyezik Y várható értékével, szórása azonban kisebb, hisz az X által magyarázott varianciától a regressziós modell megszabadított minket (nem ingyen persze, mert ismernünk kell a bétát).

4.3.2. Példa (Feltételes várható érték módszere)**.** Tegyük fel, hogy az Y valószínűségi változó értékkészlete véges sok $R_1 \cup R_2 \cup \cdots \cup R_M$ diszjunkt részhalmazra bomlik, ahol a $\mathbb{P}(Y \in R_j)$ valószínűségeket pontosan ismerjük. Legyen $y_1, y_2, \ldots, y_m$ független megfigyelések az Y változóból, ahol m_j darab megfigyelés esik az R_j halmazba. Ekkor:

$$\mu \approx \frac{\sum_{j=1}^{m} y_j}{m} = \frac{\sum_{k=1}^{M} \sum_{y_j \in R_k} y_j}{m} = \frac{\sum_{k=1}^{M} m_k \frac{\sum_{y_j \in R_k} y_j}{m_j}}{m_1 + m_2 + \cdots + m_M} = \sum \frac{m_k}{m_1 + m_2 + \cdots + m_M} \bar{y_k}$$

ahol $\bar{y_k} = \frac{\sum_{y_j \in R_k} y_j}{m_k}$ az Y mintaátlaga az $Y \in R_k$ feltétel mellett. Ez azért jó, mert a $\frac{m_k}{\sum m_j}$ hányadosok éppen az $Y \in R_k$ események valószínűségeinek becslései, melyeket a feltevés szerint ismerünk, így a képletben ezeket helyettesíthetjük pontos értékükkel:

$$\mu \approx \frac{\sum_{j=1}^{m} y_j}{m} = \sum_{k} \mathbb{P}(Y \in R_k) \cdot \bar{y_k}$$

ami tulajdonképpen a teljes várható érték tétele, így a várható érték becslését visszavezettük feltételes várható értékek becslésére. A μ-re kapott becslésünk már azáltal sokat javul, hogy a relatív gyakoriságokat az egzakt valószínűségekre cseréljük, az igazán nyerő észrevétel azonban a következő: ha a mintából csak a feltételes valószínűségeket becsüljük, az m_k/m relatív gyakoriságokat pedig nem használjuk semmire, akkor nem szükséges, hogy azok a valódi $\mathbb{P}(Y \in R_k)$ valószínűségeket (melyek pontos értékeit ismerjük) közelítsék. Ha például Y valamelyik R_k halmazon konstans, akkor egyetlen mintaelem is tökéletes becslést ad a feltételes várható értékre,

helyette több elemet generálhatunk valamelyik másik feltételes eloszlásból. Sőt általában is, ha valamelyik feltételes eloszlásnak kicsi a szórása, abból generálhatunk egy kis elemszámú mintát, cserébe, ahol a szórás nagyobb ott használhatunk nagyobbat. Ezt a módszert *rétegzés*nek nevezik. Könnyű megmutatni, hogy az optimális rétegzést (amikor a teljes minta szórása minimális) úgy kapjuk, ha az m_j számokat épp a feltételes szórásnégyzetek arányában osztjuk szét.

Varianciacsökkentő eljárásokról bővebben számos statisztika könyvből lehet informálódni, de például Botev és Ridder (2017) egy elég jó és tömör összefoglalóval szolgál. A Shapley-érték becslésére a rétegzést Maleki et al. (2013) javasolja először és Castro et al. (2017), fejleszti tovább. Shapley és Mann szavazási játékokra javasolt heurisztikáinak nagyfokú, tetszőleges TU-játékra alkamazható általánosítását Illés és Kerényi (2019) javasolja. A fejezet további részeiben ezt ismertetjük. További varianciacsökkentő módszerek Shapley-értékre való alkalmazásának, alkalmazhatóságának kutatásáról, nincs tudomásunk.

4.4. A Shapley-érték becslése ergodikus mintából

Ebben a fejezetben ismertetjuk az Illés és Kerényi (2019) által javasolt módszert, mely arról kapta a nevét, hogy a generált minta, amiből becsülni fogjuk a Shapley-értéket, nem független, de továbbra is igaz rá a nagy számok törvénye. Ez biztosítja a konzisztenciát, azaz, ha a minta elemszáma minden határon túl nő, akkor a becslésünk közelít az igazi Shapley-értékhez. Először ennek elméleti hátterét építjük föl, utána ismertetjük az algoritmust, végül a numerikus szimuláció eredményeit.

4.4.1. Elméleti háttér

4.4.1. Definíció. *Legyen $X_1, X_2, \ldots, X_i, \ldots$ azonos eloszlású, véges $\mu = E(X_j)$ várható értékű valószínűségi változók sorozata. Azt mondjuk, hogy a sorozat ergodikus, ha igaz rá a nagy számok erős törvénye, azaz*

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{\sum_{i=1}^{n} X_i}{n} = \mu\right) = 1.$$

Megjegyzés. Az idősorok elméletében az ergodicitást nem így szokás definiálni, itt az egyszerűség kedvéért azt a tulajdonságot definiáljuk, amire szükségünk lesz.

Az ergodicitás tehát egy a függetlenségnél gyengébb feltétel, mely biztosítja hogy a mintaátlagot használhassuk a várható érték becslésére. Nem független, de ergodikus sorozatot a következő általános konstrukcióval kaphatunk:

Legyen $(\Omega_1, \mathcal{A}_1, \mathbb{P}_1)$ klasszikus valószínűségi mező, azaz, $|\Omega_1|$ véges halmaz, $\mathcal{A}_1 = 2^{\Omega_1}$ a teljes hatványhalmaz, $\mathbb{P}_1(E) = \frac{|E|}{|\Omega_1|}$ $\forall E \subseteq \Omega_1$-re, és legyen $X : \Omega_1 \to \mathbb{R}$ tetszőleges valószínűségi változó μ várható értékkel. Legyen $(\Omega, \mathcal{A}, \mathbb{P})$ az Ω_1 megszámlálhatóan végtelen hatványa (a mértékterek szorzatának szokásos kontrukciója szerint), és legyen minden $j \in \mathbb{N}$ és $\omega = (\omega_1, \omega_2, \dots) \in \Omega$-re $X_j(\omega) = X(\omega_j)$. Vegyük észre, hogy az X_j-k független, az eredeti X-el azonos eloszlású valószínűségi változók. Legyen továbbá $t : \Omega_1 \leftrightarrow \Omega_1$ kölcsönösen egyértelmű transzformáció Ω_1-en és legyen $T : \Omega \leftrightarrow \Omega$ a t pontonkénti kiterjesztése Ω-ra, azaz $T(\omega) = T(\omega_1, \omega_2, \dots) = (t(\omega_1), t(\omega_2), \dots)$. Végül legyen $K \in \mathbb{N}$ egy pozitív egész paraméter, és tekintsük a következő sorozatot:

- $\forall\, l \geq 1$-re legyen $Y_{1,l} = X_l$
- minden $2 \leq k \leq K$-ra és $l \geq 1$-re legyen $Y_{k,l} = Y_{k-1,l} \circ T = Y_{1,l} \circ \underbrace{T \circ T \cdots \circ T}_{k-1} = Y_{1,l} \circ T^{k-1}$
- Álljon az Y_j sorozat az $Y_{k,l}$ változókból „soronként”,

$$Y_{1,1}, Y_{2,1}, \dots Y_{K,1}, Y_{1,2}, Y_{2,2}, \dots, Y_{K,2}, Y_{1,3}, Y_{2,3}, \dots$$

  sorrendben felsorolva őket, azaz $Y_{(l-1)K+k} = Y_{k,l}$.

Az Y_j változókra a továbbiakban szimpla és dupla indexekkel ellátva is fogunk hivatkozni, de ezek mindig ugyanazt a sorozatot jelentik.

4.4.1. Tétel. *A fent definiált Y_j sorozat minden $K \geq 2$ és $t : \Omega_1 \leftrightarrow \Omega_1$ transzformáció esetén ergodikus.*

Bizonyítás. Minden $C = (C_1, C_2, \dots) \in \mathcal{A}$ cilinderhalmaz mértéke $\mathbb{P}(C) = \left(\frac{1}{|\Omega_1|}\right)^{|\{j | C_j \neq \Omega_1\}|}$. Mivel t szürjektív $C_j = \Omega_1 \Leftrightarrow t(C_j) = \Omega_1$, így $\mathbb{P}(C) = \mathbb{P}(T(C))$, azaz T mértéktartó (ezt elegendő a cilinderhalmazokon igazolni). Először megmutatjuk, hogy a változók azonos eloszlásúak:

$$\begin{aligned}\mathbb{P}(Y_{k,l} \in B) = \mathbb{P}\left(\{\omega \in \Omega | \omega \in Y_{k,l}^{-1}(B)\}\right) = \mathbb{P}\left(\{\omega \in \Omega | T^{k-1}(\omega) \in Y_{1,l}^{-1}(B)\}\right) = \\ = \mathbb{P}\left(\{T^{k-1}(\omega) | T^{k-1}(\omega) \in Y_{1,l}^{-1}(B)\}\right) = \mathbb{P}\left(\{\omega \in \Omega | \omega \in Y_{1,l}^{-1}(B)\}\right) = \\ = \mathbb{P}(Y_{1,l} \in B) = \mathbb{P}(X_l \in B) = \mathbb{P}(X_1 \in B)\end{aligned}$$

minden $B \subseteq \mathbb{R}$ Borel halmazra. A harmadik egyenlőségnél használjuk, hogy T mértéktartó, a többi egyszerű átalakítás. Most rátérünk az ergodicitás bizonyítására. Ehhez először tegyük fel, hogy m osztható K-val, legyen $m = K \cdot L$ és tekintsük a következő átalakítást:

$$\frac{\sum Y_i}{m} = \frac{Y_1 + Y_2 + \dots Y_m}{m} =$$

$$= \frac{(Y_{1,1} + Y_{2,1} + \cdots + Y_{K,1}) + (Y_{1,2} + \cdots + Y_{K,2}) + \cdots + (Y_{1,L} + \cdots + Y_{K,L})}{KL} =$$
$$= \frac{\frac{(Y_{1,1}+Y_{1,2}+\cdots+Y_{1,L})}{L} + \frac{(Y_{2,1}+Y_{2,2}+\cdots+Y_{2,L})}{L} + \frac{(Y_{K,1}+Y_{K,2}+\cdots+Y_{K,L})}{L}}{K} = \frac{Z_1 + Z_2 + \cdots + Z_K}{K}$$

Elég megmutatni, hogy $\forall 1 \leq k \leq K$ esetén az $Y_{k,1}, Y_{k,2}, \ldots, Y_{k,L}$ változók függetlenek. Ezesetben $\mathbb{P}(Z_k \to \mu) = 1$, így $\mathbb{P}\left(\bigcap_{k=1}^{K} \{\omega \in \Omega | Z_k(\omega) \to \mu\}\right) = 1$. A Z_k-k átlaga pedig nyilván ugyanoda tart, ahová ők külön-külön.

Megjegyzés. Az $Y_{k,1}, Y_{k,2}, \ldots, Y_{k,L}$ változók függetlensége azt is jelenti, hogy $\sqrt{L}(Z_j - \mu)$ minden j-re asszimptotikusan normális eloszlású. Ez a Z_j-k átlagára nem feltétlenül igaz, például elfajuló esetben tökéletesen korreláltak is lehetnek és ilyenkor az átlaguk konstans (ami itt szerencsés eset, mert akkor pontosan megkaptuk a Shapley-értéket), de a „tipikus" esetben azt várjuk, hogy ennek a határeloszlása is normális.

Az $Y_{k,1}, Y_{k,2}, \ldots, Y_{k,L}$ változók definíció szerint függetlenek $k = 1$-re. A $k \geq 2$ esetben pedig

$$\mathbb{P}\left(\bigcap_{j=1}^{L} \{Y_{k,j} \in B_j\}\right) = \mathbb{P}\left(\left\{\omega \in \Omega | \omega \in Y_{k,1}^{-1}(B_1), \omega \in Y_{k,2}^{-1}(B_2), \ldots, \omega \in Y_{k,L}^{-1}(B_L)\right\}\right) =$$
$$= \mathbb{P}\left(\left\{\omega \in \Omega | T^{k-1}(\omega) \in Y_{1,1}^{-1}(B_1), T^{k-1}(\omega) \in Y_{1,2}^{-1}(B_2), \ldots, T^{k-1}(\omega) \in Y_{k,L}^{-1}(B_L)\right\}\right) =$$
$$= \mathbb{P}\left(\left\{T^{k-1}(\omega) | T^{k-1}(\omega) \in Y_{1,1}^{-1}(B_1), T^{k-1}(\omega) \in Y_{1,2}^{-1}(B_2), \ldots, T^{k-1}(\omega) \in Y_{k,L}^{-1}(B_L)\right\}\right) =$$
$$= \mathbb{P}\left(\left\{\omega \mid \omega \in Y_{1,1}^{-1}(B_1), \omega \in Y_{1,2}^{-1}(B_2), \ldots, \omega \in Y_{k,L}^{-1}(B_L)\right\}\right) =$$
$$= \prod_{j=1}^{L} \mathbb{P}\left(\left\{\omega \in \Omega \,|\, \omega \in Y_{1,j}^{-1}(B_j)\right\}\right) = \prod_{j=1}^{L} \mathbb{P}\left(Y_{1,j} \in B_j\right) = \prod_{j=1}^{L} \mathbb{P}\left(Y_{k,j} \in B_j\right).$$

fennáll minden $B_1, B_2, \ldots, B_L \subseteq \mathbb{R}$ Borel-halmazra. Itt a harmadik sorban T mértéktartóságát használtuk, az utolsó egyenlőség pedig azért van, mert (ahogy azt korábban bizonyítottuk), a változók azonos eloszlásúak. Most tegyük fel, hogy $K \nmid m$, és legyen $s = s(m) = \max\{j \leq m \,:\, K | j\}$ a K legnagyobb, m-nél nem nagyobb többszöröse. Ekkor

$$\frac{Y_1 + Y_2 + \ldots Y_m}{m} = \frac{Y_1 + Y_2 + \ldots Y_s}{m} + \frac{Y_{s+1} + \cdots + Y_m}{m} =$$
$$= \underbrace{\frac{Y_1 + Y_2 + \ldots Y_s}{s}}_{\downarrow \ \mu} \cdot \underbrace{\frac{s}{m}}_{\downarrow \ 1} + \underbrace{\frac{Y_{s+1} + \cdots + Y_m}{m}}_{\downarrow \ 0}$$

Itt K egy konstans melyre $m - s \leq K$, így, ha $m \to \infty$ akkor s is, ezért az első tag μ-höz tart, míg $\frac{s}{m} \to 1$. Az utolsó tag majorálható $\frac{K \cdot \max|X|}{m} \to 0$ -val, amivel kész is a bizonyítás. $\square$

4.4.2. Az algoritmus leírása

Noha a 4.4.1 tétel megfogalmazása igen absztrakt és bonyolultnak tűnik, valójában nagyon egyszerű a benne szereplő mintát generálni. Legyen $\Omega_1 = \mathcal{O}(N)$ a játékosok N halmazának összes lehetséges sorrendjeiből álló halmaz, míg X valamely $i \in N$ játékos határ-hozzájárulása. Az algoritmus két lépcsőben működik. A 4.4.1 tétel szerint szükségünk van egy $K \in N$ számra, és egy olyan $t : \Omega_1 \leftrightarrow \Omega_1$ transzformációra, mely kölcsönösen egyértelmű az Ω_1 halmazon, emellett lehetőleg gyorsan kiszámolható. Az algoritmus első lépésben egy ilyen transzformációt keres, hogy hogyan, arra később visszatérünk, majd második lépésben ezt felhasználja a Shapley-érték becslésére. Az egyszerűség kedvéért először ezt a második lépést tárgyaljuk.

Második lépés

Ha a 4.4.1 tételben szereplő t transzformációt már valahogy kiszámoltuk vagy megsejtettük, akkor a következőképpen használhatjuk fel:

- Legyen $m_2 \in N$ pozitív egész paraméter és generáljunk m_2 darab független $\omega \in \Omega_1$ sorrendet.
- Minden, az előző pontban kapott ω sorrendhez számítsuk ki a további $K-1$ darab $t^j(\omega)$: $1 \leq j \leq K-1$ sorrendet (ahol a t^j kitevő a t transzformáció j-szeri egymás utáni alkalmazását jeleni.
- Az összes így kapott Km_2 darab sorrend esetén számítsuk ki az i játékos határ-hozzájárulását, majd ezeket átlagoljuk ki, így kapjuk a Shapley-érték becslését.

Illusztrációként az algoritmus működésére, tekintsük azt az 51 játékosból álló szavazási játékot, amelyben a játékosok súlyai:

$$w = (45, 41, 27, 26, 26, 25, 21, 17, 17, 14, 13, 13, 12, 12, 12,$$
$$11, \underbrace{10, \ldots, 10}_{4}, \underbrace{9, \ldots, 9}_{4}, 8, 8, \underbrace{7, \ldots, 7}_{4}, \underbrace{6, \ldots, 6}_{4}, 5, \underbrace{4, \ldots, 4}_{9}, \underbrace{3, \ldots, 3}_{7}),$$

a kvóta pedig $q = \frac{\sum w_i}{2} + 1 = 270$. A játék az *Electoral College*, az Egyesült Államok elnökválasztási rendszerének modellje a '70-as évek súlyaival, melynek Shapley-értékét ma már pontosan ismerjük, így kiválóan használható az algoritmusok tesztelésére (ennek egy régebbi verzióját használta Mann és Shapley (1962) is). A legnagyobb súlyú játékos, California Shapley-értékét szeretnénk megbecsülni.

Az algoritmushoz a $t : x \mapsto (x + 17) \pmod{51}$ transzformációt fogjuk használni, mely harmadrendű, tehát $K = 3$ lesz így végeredményben m_2-ször ismételjük a következőket:

- Generálunk egy véletlen sorrendet és kiszámojuk az $i = 1$ játékos határ-hozzájárulását, jelölje ezt Y_1.
- Tekintsük most azt a sorrendet, amit úgy kapunk az előzőből, hogy az utolsó 17 játékos egymás közti relatív sorrendjének megőrzése mellett beáll a sor elejére (a többi játékos is megőrzi egymás közti relatív sorrendjét, de mindenki 17 játékossal hátrébb kerül). Jelölje Y_2 az első játékos határ-hozzájárulását ennél a sorrendnél.
- Ismételjük az előző pontot, így kapjuk az Y_3 változót. (Ha ugyanezt harmadszor is megismételnénk, akkor negyediknek visszakapnánk az eredeti sorrendet és Y_1-et.)

Az algoritmust egy megegyező elemszámú ($3m_2$) független mintavételezéssel összehasonlítva, az eredményeket a 4.1 táblázat tartalmazza. A szórást 1000 elemű szimuláció alapján becsültük. A módszer kb. 10%-al csökkenti a Shapley-érték becslésének szórását.

Minta elemszáma	$m = 10^5 - 1$	$m = 10^6 - 1$
Független minta szórása	0.0008568	0.0002890
Algoritmus szórása	0.0008012	0.0002609
A két szórás aránya	0.9351519	0.9027682

4.1. táblázat. Ergodikus minta tesztelése az amerikai elnökválasztás példáján.

Ezt elméleti alapon is beláthatjuk. Jelölje ideiglenesen p_1 az egyes számú játékos Shapley-értékét, mely egyébként kb. 0,088. A játékosok egy véletlen sorrendjében pontosan egy olyan játékos van, aki átbillenti a szavazást, az ő határ-hozzájárulása 1, mindenki másé 0, p_1 éppen annak a valószínűsége, hogy ez a játékos éppen California lesz. Ha egy adott véletlen sorrendnél California épp jókor érkezik, hogy a „mérleg nyelve" legyen, az azt jelenti, hogy egy olyan koalícióhoz csatlakozott, amelyben a súlyok összege kisebb, mint a q kvóta, de nagyobb, mint $q - 45$. Ha most őt 17-el hátrébb küldjük a sorban, úgy, hogy az eredetileg 17 utolsó játékos beelőzi, akkor ez a helyzet már nem állhat fönt, mert a 17 leggyengébb játékos szavazóereje is 62, ezért a nyerő koalíció korábban fog kialakulni. Hasonlóan könnyű meggondolni, hogy az sem lehetséges, hogy eredetileg az utolsó 17 játékos között érkezett, és 1 volt a határ-hozzájárulása, majd átkerült az első 17 játékos közé és ennél a sorrendnél is 1 maradt. Vagyis, ha egy olyan sorrendet generálunk, ahol California határ-hozzájárulása 1, akkor a másik kettőnél már biztosan 0 lesz, azaz Y_1, Y_2 és Y_3 közül legfeljebb az egyik lehet nem nulla. Mindhárom Y_j bináris eloszlású és várható értékük p_1. Két ilyen változó kovarianciájának minimális értéke $-p_1^2$, és ezt a minimumot pontosan akkor kapjuk, ha a két változó szorzata azonosan nulla. A Y_j-k tehát, nem csak hogy páronkként negatívan korrelálnak (s így az összegük szórása kisebb, mint két

ugyanilyen eloszlású független változóé lenne), hanem a lehető „legnegatívabban” korrelálnak, így az összegük szórása eléri az elméleti minimumot. A változók kovaraianciamátrixa tehát:

$$\begin{bmatrix} p_1(1-p_1) & -p_1^2 & -p_1^2 \\ -p_1^2 & p_1(1-p_1) & -p_1^2 \\ -p_1^2 & -p_1^2 & p_1(1-p_1) \end{bmatrix}$$

A független változók kovaraianciamátrixa ugyanez lenne, csak a nem diagonális elemei csupa nullák. Három ilyen változó átlagának szórásnégyzete épp a kovaraianciamátrix elemeinek összege 9-el osztva, vagyis az $\frac{Y_1+Y_2+Y_3}{3}$ változó, és három ugyanilyen eloszlású de független változó szórásának aránya $\sqrt{\frac{3p_1(1-p_1)-6p_1^2}{3p_1(1-p_1)}} = \sqrt{1-2\cdot\frac{p_1}{1-p_1}}$. Ez $p_1 = \frac{1}{3}$ esetén éppen eliminálná a szórást[26], jelen esetben, $p_1 \approx 0,088$ esetén ennek értéke kb. 0,89, ennyied részére csökkenti az algoritmus a becslési hibát.

Első lépés

Most rátérünk arra a kérdésre, hogy hogyan találhatunk egy megfelelő transzformációt az algoritmushoz. Jegyezzük meg, hogy a 4.4.1 tétel a t transzformációtól semmit nem követel meg, azon kívül, hogy legyen kölcsönösen egyértelmű, és ebből kifolyólag mértéktartó, azonban, ha a transzformáció nem negatívan korreláló mintákat eredményez, akkor nem lesz hasznos, mert akkor az algoritmus nem csökkeni, hanem növeli a Shapley-érték becslési hibáját. Egy példát már láttunk megfelelő transzformációra, azonban azt a játékhoz ad-hoc találtuk ki (intuíció alapján, emberi képességekkel), a cél most az, hogy ezt az algoritmus valahogy saját maga konsturálja meg. Ez a probléma teljes általánosságban reménytelenül nehéznek illetve számításigényesnek tűnik, egyelőre csak a $K = 2$ speciális esetre tudunk ilyen módszert bemutatni.

Jelölje Π_n az $\{1, 2, \ldots, n\}$ halmazon ható szimmetrikus csoportot. Minden $\pi \in \Pi_n$ természetes módon meghatároz egy sorrendeken ható $t_\pi : \mathcal{O}(N) \to \mathcal{O}(N)$ transzformációt. Legyen ugyanis minden $o \in \mathcal{O}(N)$-re és $\pi \in \Pi_n$-re $\forall i \in N$-re $t_\pi(o)(i) = (\pi \circ o)(i) = \pi(o(i))$. Ez teljesen értelmes jelölés, mert a 1.2.4. Definíció szerint egy sorrend értékkészlete pont az a halmaz, ahol a szimmetrikus csoport hat, és kölcsönösen egyértelmű függvények kompozíciója is kölcsönösen egyértelmű. Ha például egy $\pi \in \Pi_n$-re $\pi(j) = k$ (ahol $1 \leq j, k \leq n$ egész számok), ez azt jelenti, hogy a játékosok minden $o \in \mathcal{O}(N)$ sorrendjéhez egy olyan $t_\pi(o)$ sorrendet rendelünk, ahol az eredetileg j-ediknek értkező játékos most k-adiknak fog érkezni. Az összes $t : \mathcal{O}(N) \to \mathcal{O}(N)$ transzformációk száma $|\mathcal{O}(N)|! = n!!$, míg $|\Pi_n| = n!$, így a legtöbb transzformáció ilyen alakban nem kapható meg, csak azok, ahol, az, hogy egy játékos hová kerül a transzformáció után, nem függ attól, hogy ő kicsoda, csak attól, hogy eredetileg hányadiknak érkezett. Shapley és Mann 4.2.1. példában látott két intuitíve nagyon hasonló módszerének B verziója megkapható ilyen

[26] $p_1 > 1/3$-ra ilyen változók nem léteznek.

alakban (π épp az n hosszú ciklikus permutáció), az A verzió azonban nem. Hogy ezzel mennyit veszítünk, azt nem tudjuk, de az így megkapható transzformációk száma is túl nagy ahhoz, hogy mindet végignézzük, csak olyanok között fogunk keresgélni, melyekre $\pi^2 = \mathrm{id}$ a Π_n csoport egységeleme, azaz melyek előállnak diszjunkt transzpozíciók szorzataként. Ez szemléletesen azt jelenti, hogy párba állítjuk az $\{1, 2, \ldots, n\}$ halmaz elemeit (lehetnek elemek, amelyek saját magukkal állnak párban) és a párokat felcseréljük. Azt, hogy hogyan érdemes a párokat kialakítani, statisztikai alapon döntjük el. Legeneráljuk a játékosok „néhány" lehetséges sorrendjét, minden játékos párt felcserélünk, kiszámoljuk ezen sorrendekhez tartozó határ-hozzájárulásokat, megnézzük, mely két játékos felcserélése esetén fognak ezek minél inkább negatívan korrelálni, majd ezeket a párokat összekombináljuk. Legyen tehát m_1 egy tetszőleges természetes szám $i \in N$ egy tetszőleges (de rögzített) játékos, és tekintsük a következő procedúrát:

- Generáljunk m_1 darab lehetséges sorrendet a játékosok N halmazán, legyenek ezek: $s_1, s_2, \ldots, s_{m_1}$, és jelölje $\mathcal{S}$ az i játékos határ-hozzájárulásaiból álló vektort, ez egy m_1 hosszú adatsor.

- Minden $a, b \in \{1, 2, \ldots, n\}$ pár esetén legyen $s_1^{(a,b)}, s_2^{(a,b)}, \ldots, s_{m_1}^{(a,b)}$ a játékosok azon sorrendje, melyet úgy kapunk, hogy a megfelelő s_j-ben az a-adiknak és b-ediknek érkező játékost felcseréljük. Itt az a-adik és b-edik játékos felcserélése egy (speciális) transzformáció: $s_j^{(a,b)} = \langle a, b \rangle \circ s_j$ ahol $\langle a, b \rangle \in \Pi_n$ az a és b elemek felcserélése.

- Jelölje a $S_{(a,b)}$ az i játékos határ-hozzájárulásaiból álló vektort az $s_1^{(a,b)}, s_2^{(a,b)}, \ldots, s_{m_1}^{(a,b)}$ sorrendek mellett. Ez tehát minden (a, b) pár esetén egy m_1 elemű vektor.

- Legyen minden (a, b)-re $C_{(a,b)}$ az *eredeti* S és az $S_{(a,b)}$ adatsorok kovarianciája.[27]

- A $C_{(a,b)}$ számok az $\{1, 2, \ldots, n\}$ alaphalmazon, mint csúcshalmazon értelmezett teljes gráf éleinek egy súlyozásaként is tekinthetőek. Keressünk mohó algoritmussal ebben a gráfban egy minimális súlyú teljes párosítást, és feleltessük meg egymásnak azokat a pontokat, melyeket a párosítás összeköt. Ez részletesebben kiírva a következőt jelenti: Legyen $E \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ kezdetben az üres halmaz.

  1. Tekintsük az $(a, b) \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ párok $C_{(a,b)}$ szerinti növekvő sorrendbe rendezését.

  2. Adjuk hozzá E-hez a sorozat első elemét.

  3. Húzzuk ki a sorozatból az összes olyan (a, b) elemet, melyre akár a akár b szerepel valamelyik E-beli párban (bármilyen sorrendben).

[27] Ha m_1 elég nagy, akkor számolhatnánk a korrelációs együtthatóval is, de így egyszerűbb, mert biztosan elkerüljük a 0-val való osztást.

Eredeti sorrend	Csere(1,2)	Csere(1,3)	Csere(1,4)	Csere(2,3)	Csere(2,4)	Csere(3,4)
CBAD (2)	BCAD (2)	ABCD (0)	DBAC (2)	CABD (0)	CDAB (2)	CBDA (10)
DABC (0)	ADBC (0)	BADC (0)	CABD (0)	DBAC (2)	DCBA (10)	DACB (0)
BDAC (2)	DBAC (2)	ADBC (0)	CDAB (2)	BADC (0)	BCAD (2)	BDCA (10)
BACD (0)	ABCD (0)	CABD (0)	DACB (0)	BCAD (2)	BDCA (10)	BADC (0)
ABDC (0)	BADC (0)	DBAC (2)	CBDA (10)	ADBC (0)	ACDB (0)	ABCD (0)
BADC (0)	ABDC (0)	DABC (0)	CADB (0)	BDAC (2)	BCDA (10)	BACD (0)
DCBA (10)	CDBA (10)	BCDA (10)	ACBD (0)	DBCA (10)	DABC (0)	DCAB (2)
BCAD (2)	CBAD (2)	ACBD (0)	DCAB (2)	BACD (0)	BDAC (2)	BCDA (10)
ABCD (0)	BACD (0)	CBAD (2)	DBCA (10)	ACBD (0)	ADCB (0)	ABDC (0)
DBCA (10)	BDCA (10)	CBDA (10)	ABCD (0)	DCBA (10)	DACB (0)	DBAC (2)

4.2. táblázat. Párosítás keresésének algoritmusa

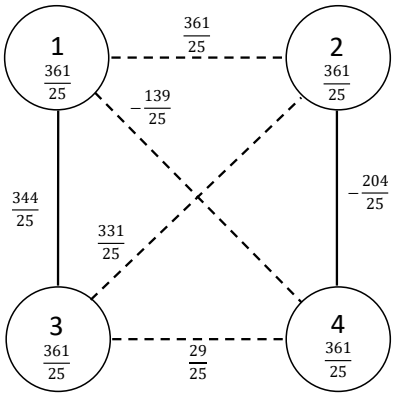
4. Ha nem marad több elem a sorozatban, kész vagyunk, egyébként ugorjunk a 2.) pontra.

Most legyen π az a permutáció, mely minden $a \in \{1, 2, \ldots, n\}$ számot felcserél azzal az egyértelműen létező b számmal, melyre (a, b) vagy (b, a) E-ben van. Ezzel kész vagyunk.

Megjegyzés. A mohó algoritmus természetesen nem alkalmas a minimális súlyú teljes párosítás megkeresésére, de annak egy elfogadható közelítését adja. Valódi teljes párosítást is kereshetnénk a magyar módszerrel, de annak nagyobb a műveletigénye, és a véges mintából becsült kovarianciákban lévő zaj miatt az algoritmust érdemben valószínűleg nem javítaná.

A két lépés összekapcsolása

A teljes algoritmus működését a következő egyszerű játékon illusztráljuk: a négy játékos A, B, C, és D, minden háromelemű koalíció 2-t ér a nagykoalíció értéke 12, az összes többi koalíció értéke pedig 0. Az A játékos Shapley-értékét szeretnénk megbecsülni. Az algoritmus először legenerál a négy játékosból $m_1 = 10$ darab véletlen sorrendet, ezeket látjuk a 4.2. táblázat első oszlopában. Ezután minden lehetséges módon felcseréli valamelyik két játékost, ezzel $m_1 \cdot \binom{n}{2}$ darab új sorrendet kap, melyek a táblázat további hat oszlopában láthatóak. A táblázatban zárójelben feltüntettük minden egyes sorrend mellett az A játékos határ-hozzájárulását is. Ezután az algoritmus a 4.1. ábrán lévő gráfban keres egy párosítást, melyben az a és b él $(1 \leq a \leq b \leq 4)$ súlya a 4.2. táblázat Csere(a,b) fejlécű és első oszlopában lévő határ-hozzájárulások kovarianciája (illetve a csúcsokban lévő szám az első oszlop szórásnégyzete). Az algoritmus először kiválasztja a $-204/25$ súlyú $(2, 4)$ élet, és ezután már nem foglalkozik egyetlen olyan éllel sem, amely a 2-es vagy 4-es csúcsra illeszkedik. Két lehetőség marad tehát, vagy kiválasztjuk az (1,3) élet is, vagy ez a két csúcs önmaga párja lesz. Mivel $344/25 < 361/25$, ezért az előbbit választjuk, tehát az 1-es és 3-as csúcs is párban lesz. Az eredményt úgy kell értelmezni, hogy



4.1. ábra. Az optimális transzformáció keresése az ergodikus minta generálásához.

az algoritmus a második lépéshez azt a t transzformációt fogja használni, mely a játékosok egy (i_1, i_2, i_3, i_4) sorrendjéhez az (i_3, i_4, i_1, i_2) sorrendet rendeli. Miután a megfelelő t transzformációt megtalálta, az algoritmus visszamegy a kályhához és legenerál m_2 darab új érkezési sorrendet. Az m_2 paraméter megválasztására később visszatérünk, a példa kedvéért legyenek az új sorrendek most $ADBC$, $CABD$, $BDCA$, $BACD$, $DACB$, $ABDC$, $ADBC$ és $ABDC$. Ez egy elég szerencsétlen minta, mert az A játékos határ-hozzájárulása mindegyik sorrendnél 0, kivéve a harmadikat, ahol 10, így a Shapley-értékre kapott becslésünk 5/4. Az algoritmus most alkalmazza a t transzformációt mindegyik sorrendre és kap újabb m_2 darab sorrendet, felcserélve a másodiknak érkező játékost a negyediknek érkezővel, és a harmadiknak érkezőt az elsőnek érkezzővel: $BCAD$, $BDCA$, $CABD$, $CDBA$, $CBDA$, $DCAB$, $BCAD$, illetve $DCAB$. A transzformációval azt értük el, hogy ahol eredetileg 10 volt a határ-hozzájárulás, ott most 0 lett, ahol eredetileg 0 volt, ott most 2 vagy 10 lett, amivel kaptunk egy az előzőhöz hasonlóan rossz becslést a Shapley-értékre, ami 38/8. Ha azonban a kettőt kiátlagoljuk, az eredmény éppen 3, ami hajszál pontosan a korrekt érték. A két rossz becslésből azért kaptunk egy jót, mert a két adatsor negatívan korrelál, így ha az egyik nagyon mellélő az egyik oldalon, akkor a másik nagy valószínűséggel kompenzálja azzal, hogy a másik oldalon lő mellé.

4.4.3. Komplexitás

Legyen a játék komplexitása (a karakterisztikus függvény kiértékelésének műveletigénye) G_n és vizsgáljuk meg az algoritmus lépésszámát! Vegyük sorba a első lépésben végrehajtott utasításokat:

1. Legenerálunk m_1 véletlen sorrendet: ez lényegében egy $m_1 \times n$ -es mátrix, $O(m_1 n)$ mű-

velet.

2. Cserélgetjük a játékosokat, azaz, az előző lépésben létrehozott mátrix oszlopait. Ehhez értelemszerűen nem másoljuk le újra az egész mátrixot, csak kicserélünk két oszlopot, aztán vissza és megint kicserélünk két másikat és így tovább, egy csere m_1-ben lineáris mennyiségű művelet, amit $\binom{n}{2}$-ször végzünk el, tehát összesen $O(m_1 n^2)$ művelet.

3. Az eredeti, és az oszlopok cserélgetésével kapott mátrixra is kiszámolunk összesen $O(m_1 n^2)$ határ-hozzájárulást. Ennek műveletigéne $O(m_1 n^2 G_n)$.

4. Kiszámoljuk a kovarianciákat, ez $O(m_1 n^2)$ művelet.

5. Rendezés: $O(n^2 \log(n))$ művelet.

6. Párosítás mohó algoritmussal: könnyen látható, hogy ez megvalósítható a kovarianciák számában lineáris számú, azaz $O(n^2)$ művelettel.

Ha élünk azzal a nem túl erős megszorítással, hogy $G_n = \Omega(n)$[28] akkor láthatjuk, hogy itt messze a legnagyobb műveletigényű lépés a 3.) pontban a határ-hozzájárulások kiszámítása, az összes többi elhanyagolható. Ezt a tagot tehát érdemes kicsit pontosabban megbecsülni. Összesen $m_1\binom{n}{2}$ határ-hozzájárulást kell kiszámolni. Vegyük észre, hogy némi számolást meg tudunk spórolni. Amikor két játékost, mondjuk az a-adikat és b-ediket felcseréljük, akkor az i játékos határ-hozzájárulása nem változik, ha ő eredetileg nem az a-adik után és a b-edik előtt érkezett (vagy fordítva). Mivel az i játékos egyenletes eloszlással, bármikor érkezhet, a és b pedig végigfut 1-től n-ig, a karakterisztikus függvényt átlagban csak kb. az esetek 1/3-ában kell újra kiértékelni, az esetek kb. 2/3-ában a csere nem változtatja meg azt a koalíciót, amihez az i játékos csatlakozik. Ebből kifolyólag azt mondhatjuk, hogy az algoritmus első lépésének futásideje kb. $\frac{m_1 n^2 G_n}{6}$.
A második lépés sokkal egyszerűbb, itt m_2 darab véletlen sorrend generálása történik, ami $O(m_2 n)$ művelet, majd ezek transzformálása, ami ugyanennyi, végül mind a $2m_2$ sorrend mellett a határ-hozzájárulások kiszámítása, az egész összesen $O(m_2 G_n)$ művelet.

Mindezt azért számoltuk ki, mert szeretnénk összehasonlítani az algoritmust, az egyszerű független mintavételezéssel, és ehhez a paramétereket (a minta elemszámát: m-et, illetve m_1-et és m_2-t) úgy kell beállítani, hogy a futási idők (legalább átlagban, és elég nagy minta esetén) minél pontosabban megegyezzenek. Egy m elemű független mintával az egyszerű mintavételezés komplexitása nyilvan mG_n. Az algoritmus paramétereit tehát úgy a legigazságosabb beállítani, hogy

$$\frac{m_1 n^2 G_n}{6} + 2m_2 G_n = m_2 G_n \qquad \text{azaz} \qquad \frac{m_1 n^2}{6} + 2m_2 = m$$

[28] Ez azt jelenti, hogy a játék karakterisztikus függényének kiértékelése a játékosok számában legalább lineáris futásidejű, ami elég ésszerű feltétel, nagyon nehéz értelmes játékot elképzelni, ahol ennél gyorsabb lenne.

teljesüljön. Az m_1 és m_2 paraméterek bármely olyan választása esetén, mely teljesíti a fenti összefüggést, az algoritmus egy m elemű független mintás Monte-Carlo szimulációval megegyező[29] átlagos futásidejű, arról azonban még nem szóltunk semmit, hogy a csak egymás rovására növelhető m_1 és m_2 paramétereket hogyan határozzuk meg. Erre az eredmények elemzése után térünk vissza.

Megjegyzés. Ha a játékosok n száma nagyon nagy, akkor az algoritmus első lépésének négyzetes futásideje esetleg vállalhatatlanul lassú. Ez nem teszi feltétlenül használhatatlanná az algoritmust, ugyanis megtehetjük, hogy olyan transzformáció helyett, amely játékosokat állít párokba, olyat keresünk, amely a játékosok például $\sqrt{n}$ méretű csoportjait cserélgeti és állítja párokba. Ezzel kapcsolatban részletes eredményeket nem fogunk bemutatni, de a szimulációk alapján úgy tűnik, hogy ez a verzió nem ad rosszabb eredményt, sőt néha éppen hogy stabilabbá teszi és megkönnyíti egy jól használható transzformáció megkeresését.

4.4.4. Eredmények

Az algoritmus tesztelésére a következő nyolc konkrét játékot használtuk:

1. A 4.4.2 fejezetben bemutatott szavazási játék az amerikai elnökválasztásra.

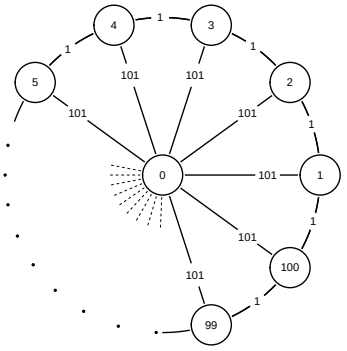
2. Szimmetrikus szavazási játék 100 játékossal (minden játékos súlya azonos, a kvóta 50).

3. Legyen $N_1 = \{1, 2, \ldots, 50\}$, $N_2 = \{51, 52, \ldots, 100\}$, $N = N_1 \cup N_2$, és legyen minden $S \subseteq N$-re $v(S) = \min(|S \cap N1|, |S \cap N_2|)$. Ez a játék a szakirodalomban eredetileg *shoes game*, magyarul *kesztyűjáték* néven ismert, minden koalíció értéke az egymást kiegészítő tárgyakból képezhető párok száma.

4. Repülőtér-játék. Legyen $N = \{1, 2, \ldots, 100\}$ és $c = (\underbrace{1, \ldots, 1}_{8}, \underbrace{2, \ldots, 2}_{12}, \underbrace{3, \ldots, 3}_{6}, \underbrace{4, \ldots, 4}_{14}, \underbrace{5, \ldots, 5}_{8}, \underbrace{6, \ldots, 6}_{9}, \underbrace{7, \ldots, 7}_{13}, \underbrace{8, \ldots, 8}_{10}, \underbrace{9, \ldots, 9}_{10}, \underbrace{10, \ldots, 10}_{10})$, és legyen a karakterisztikus függvény $v(S) = \max\{c_i | i \in S\}$. A c vektor azt írja le, mekkora költség az egyes repülőgépekhez kifutópályát építeni. Egy repülőgép tehát csak akkor növeli egy koalíció költségét, ha nagyobb, mint a koalícióban a legnagyobb gép, és csak miatta kell nagyobbra építeni a kifutópályát.

5. Legyen a játékosok halmaza $N = \{0, 1, 2, \ldots, 100\}$ és legyen tetszőleges S koalíció értéke a 4.2. ábrán lévő gráf S-beli csúcsai és a 0 csúcs által feszített részgráf minimális költségű feszítőfájának költsége.

[29] Vagy legalábbis a lehető legjobban közelíti azt.



4.2. ábra. Minimális költségű feszítőfa (MCST) játék gráfja

6. A 3.1.1 példa csődjátéka $n = 100$ játékosra. Az eszközök értéke 200, a követelések megegyeznek repülőtér-játékban lévő költségekkel.

7. A 3.1.1 példában bemutatott tartozásos játék, a paraméterei megegyeznek a csődjáték paramétereivel.

8. Legyen $N_1 = \{1, 2, \ldots, 50\}$, $N_2 = \{51, 52, \ldots, 100\}$, $N = N_1 \cup N_2$, és tegyük fel, hogy minden játékos az N_1 halmazban pontosan egy N_2-belivel van párba állítva. Legyen minden S koalíció $v(S)$ értéke az S-beli elemekből kirakható párok száma. Ez tehát hasonlít a *shoes game* játékra, de itt nem képezhető pár bármely jobb és bármely bal lábas cipőből hanem mindenkinek csak egy párja van, például képzelhetjük úgy, hogy minden pár cipő különböző színű.

A numerikus szimuláció eredményeit a 4.3. táblázat tartalmazza. Mindegyik játékra öt különböző paraméterbeállítással futtattunk egy 1000 elemű szimulációt, hogy megbecsüljük az algoritmus által becsült Shapley-érték szórását. Ezt tartalmazza a σ_E oszlop. Összehasonlításként feltüntettük az egyszerű mintavétel szórását, és az utolsó oszlopban a kettő hányadosát. A 4. és 6. játék esetében a legnagyobb súlyú utolsó játékos Shapley-értékét számoltuk, a többinél az elsőét. Az eredményeket röviden három pontban foglalhatjuk össze.

- Ha az algoritmust túl kis elemszámú mintával futtatjuk, akkor a számítási kapacitás túl nagy részét köti le a megfelelő t transzformáció megtalálása, így rosszabb eredményt produkál, mint az egyszerű mintavételezés. A táblázat ezen soraiban 1-nél nagyobb szám van az utolsó oszlopban.

- Ha az algoritmus „elég nagy” elemszámú mintát generál, akkor a nyolc esetből háromnál

(1., 2. és 4. játék) nem képes érdemi javulást elérni, és nagyon hasonló eredményt produkál, mint az egyszerű mintavételezés. A táblázat ezen sorainak utolsó oszlopában 1-hez nagyon közeli számok láthatóak.

- A nyolc esetből 5-nél az algoritmus szignifikánsan csökkenti a becslés hibáját az egyszerű mintavételezéshez képest, a legdrasztikusabban a 7. játékban (a táblázat 35. sora).

Az algoritmus első lépésben m_1 elemű minta felhasználásával egy t transzformációt keres, amelynek segítségével a második lépésben két m_2 elemű negatívan korreláló mintát tud generálni. Adott számítási kapacitás mellett, minél nagyobb az első lépésben felhasznált m_1 mintaelemszám, a második lépésben annál kisebb m_2-t lehet választani, azonban magasabb m_1 választása (egy bizonyos határig) „jobb" transzformációt, azaz kisebb (nagyobb negatív) korrelációt eredményezhet, ami viszont lehet, hogy nagyobb mértékben csökkenti a becslési hibát, mint a magasabb m_2 érték. A két paraméter között tehát egy nagyon nemtriviális trade-off van. Hogy ezt számszerűsítsük, legyen az algoritmus által generált mintapár $Y_{1,1}, Y_{1,2}, \ldots, Y_{1,m_2}$ illetve $Y_{2,1}, Y_{2,2}, \ldots, Y_{2,m_2}$ és jelölje ρ a köztük lévő korrelációt. Ha az $(Y_{1,i}, Y_{2,i})$ változók együttes eloszlása azonos, és függetlenek, akkor a korrelált minta átlagának szórásnégyzete

$$
\begin{aligned}
D^2\left(\frac{Y_{1,1,}+Y_{2,1}+Y_{1,2}+Y_{2,2}+\cdots+Y_{1,m_2}+Y_{2,m_2}}{2m_2}\right) = \\
= D^2\left(\frac{\frac{Y_{1,1}+Y_{2,1}}{2}+\frac{Y_{1,2}+Y_{2,2}}{2}+\cdots+\frac{Y_{1,m_2}+Y_{2,m_2}}{2}}{m_2}\right) = \\
= \frac{1}{m_2^2}\cdot m_2 \cdot D^2\left(\frac{Y_{1,1}+Y_{2,1}}{2}\right) = \\
= \frac{1}{m_2}\cdot\frac{1}{4}\cdot\left(D^2\left(Y_{1,1}\right)+D^2\left(Y_{2,1}\right)+2\cdot cov(Y_{1,1},Y_{2,1})\right) = \\
= \frac{1}{4m_2}\cdot\left(\sigma^2+\sigma^2+2\cdot\sigma\cdot\sigma\cdot\rho\right) = \frac{1}{4m_2}\cdot\left(2\sigma^2\cdot(1+\rho)\right) = \frac{\sigma^2}{2m_2}\cdot(1+\rho)
\end{aligned}
$$

ahol σ egy darab $Y_{i,j}$ azaz a határ-hozzájárulás szórása. A mintaátlag szórása egy m elemű független minta esetén egyszerűen $\frac{\sigma}{\sqrt{m}}$, a kettő hányadosa (azaz a 4.3. táblázat utolsó oszlopában lévő számok elméleti értéke)

$$
\frac{\sigma_E}{\sigma_R} = \frac{\frac{\sigma}{\sqrt{2m_2}}\sqrt{1+\rho}}{\frac{\sigma}{\sqrt{m}}} = \sqrt{\frac{m}{2m_2}(1+\rho)}, \tag{4.1}
$$

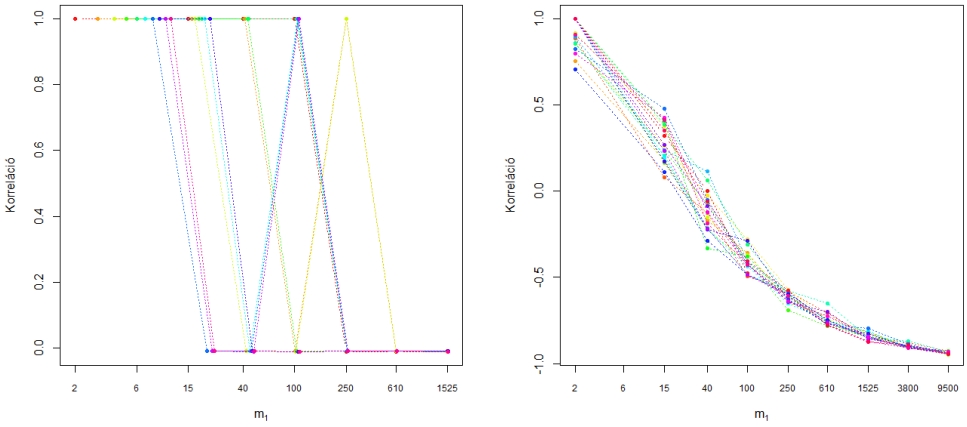
annak a szükséges és elégséges feltétele pedig, hogy az algoritmus javítsa a becslést az egyszerű véletlen mintához képest:

$$
\frac{\sigma}{\sqrt{2m_2}}\sqrt{(1+\rho)} \leq \frac{\sigma}{\sqrt{m}} \Longleftrightarrow m\cdot(1+\rho) \leq 2m_2.
$$

Ez a feltétel természetesen nem tud teljesülni, ha $\rho \geq 0$, hiszen a pozitív korreláció nem csökkenti hanem növeli a mintaátlag szórását. Ha azonban létezik olyan m_1, amellyel az algoritmus első

Játék	m	m_1	m_2	σ_R	σ_E	σ_E/σ_R
	100 000	100	28 325	0,000897	0,001201	1,34
	600 000	100	278 325	0,000366	0,000362	0,99
1.	4000 000	100	1 978 325	0,000142	0,000136	0,96
	600 000	1000	83 250	0,000366	0,000714	1,95
	4000 000	1000	1 783 250	0,000142	0,000146	1,03
	1000 000	500	83 333	0,000099	0,000237	2,39
	2000 000	50	958 334	0,000070	0,000100	1,43
2.	10 000 000	50	4 958 334	0,000031	0,000030	0,98
	2000 000	1000	166 667	0,000070	0,000170	2,43
	10 000 000	1000	4 166 667	0,000031	0,000035	1,12
	400 000	100	116 667	0,000791	0,000848	1,07
	3000 000	100	1 416 667	0,000289	0,000229	0,79
3.	10 000 000	100	4 916 667	0,000158	0,000127	0,81
	3000 000	1500	250 000	0,000289	0,000415	1,43
	10 000 000	1500	3 750 000	0,000158	0,000101	0,64
	200 000	50	58 333	0,002825	0,003602	1,28
	2000 000	50	958 333	0,000893	0,000911	1,02
4.	7000 000	50	3 458 333	0,000478	0,000465	0,97
	2000 000	500	583 333	0,000893	0,001159	1,30
	7000 000	500	3 083 333	0,000478	0,000498	1,04
	250 000	100	41 667	0,162058	0,204571	1,26
	1700 000	100	766 667	0,062146	0,052712	0,85
5.	4000 000	100	1 916 667	0,040514	0,033662	0,83
	1700 000	1000	16 667	0,062146	0,198159	3,19
	4000 000	1000	1 166 667	0,040514	0,023188	0,57
	200 000	100	16 667	0,010639	0,018708	1,76
	1000 000	100	416 667	0,004758	0,003526	0,74
6.	10 000 000	100	4 916 667	0,001505	0,001047	0,70
	1000 000	500	83 333	0,004758	0,007708	1,62
	10 000 000	500	4 583 333	0,001505	0,001051	0,70
	200 000	100	166 667	0,180138	0,244463	1,36
	1000 000	100	416 667	0,080560	0,056791	0,70
7.	10 000 000	100	4 916 667	0,025475	0,015614	0,61
	1000 000	500	83 333	0,080560	0,045475	0,56
	10 000 000	500	4 583 334	0,025475	0,006730	0,26
	250 000	100	41 667	0,001000	0,001520	1,52
	2000 000	100	916 667	0,000354	0,000278	0,79
8.	5000 000	100	2 416 667	0,000224	0,000198	0,88
	2000 000	1000	166 667	0,000351	0,000442	1,26
	5000 000	1000	1 666 667	0,000224	0,000150	0,67

4.3. táblázat. Ergodikus és független minta összehasonlítása 1000 elemű szimulácó alapján



4.3. ábra. Ergodikus minta korrelációs együtthatója a mintanagyság függvényében

lépésében talált transzformációra ρ negatív, akkor az algoritmus a

$$\sqrt{\frac{m}{2m_2}(1+\rho)} \longrightarrow \sqrt{1+\rho}$$

faktorral javítja a becslést, azaz „elég nagy" minta esetén kb. $\sqrt{1+\rho}$-szorosára csökkenti a becslés szórását a független mintához képest.

Vegyük észre, hogy ha ismernénk ρ-t, mint m_1 függvényét, akkor triviális lenne megmondani, hogy hogyan kell az algoritmust optimálisan paraméterezni, és érdemes-e egyáltalán használni. A 4.1 egyenletben - fix m mellett - m_2-t m_1-el kifejezve és négyzetre emelve, azt kapjuk, hogy az ergodikus minta és a független minta szórásnégyzetének aránya

$$H(m_1) = \frac{m}{m - \frac{m_1 n^2}{6}}\big(1+\rho(m_1)\big).$$

A szorzat első tényezője $m_1 = m \cdot \frac{6}{n^2}$-ig monoton nő, a második tényezőtől pedig, bár garancia nincs rá, azt várjuk hogy monoton fogyó legyen, így tipikus esetben egy `U` alakú függvényt kéne kapnunk, melynek egyértelműen létezik egy globális m_1^* minimuma. Az algoritmus pontosan akkor csökkenti (nem növeli) a becslés szórását a független mintához képest ha erre $H(m_1^*) \leq 1$.

Ezzel persze nem sokra megyünk, mert ρ-t nem ismerjük, sőt még csak nem is függvény, hanem valószínűségi változó, de nem reménytelen, hogy a korreláció m_1 függvényében bizonyos játékosztályokon hasonlóan viselkedik, és legalább arra találhatunk valami támpontot, mikor és milyen m mintanagyságtól kezdve lehet érdemes az algoritmust használni. A 4.3. ábra két eléggé különböző esetet illusztrál. A bal oldali grafikonon a szimmetrikus szavazási játék esetén látható a korreláció, m_1 függvényében. Ez természetesen kis elemszám esetén 1-ről indul és valahol hat és hatszáz között nullára zuhan. Abban az esetben, ha a felhasznált m_1 darab

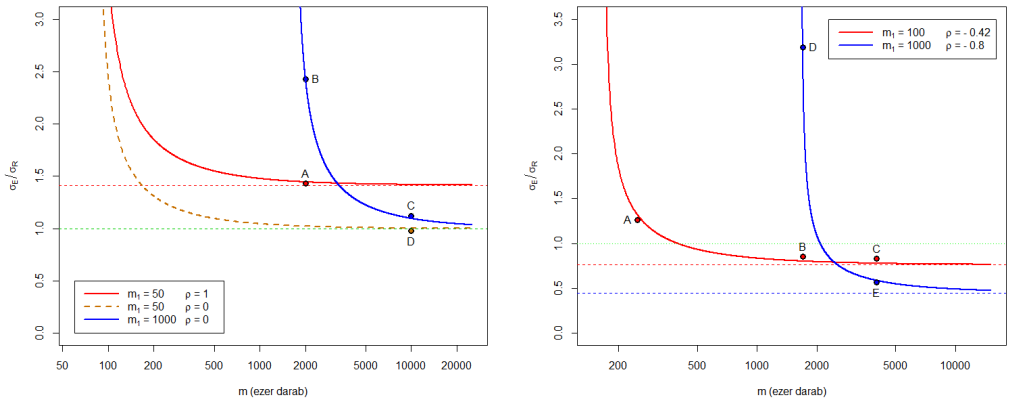
véletlen sorrend között egyetlen egy olyan sincs, amikor az első játékos 50-ediknek érkezik, a becsült Shapley-érték konstans nulla lesz, amivel természetesen semmit sem lehet kezdeni, ilyen esetben az algoritmus által talált transzformáció az identitás lesz. Ha akad olyan sorrend, ahol az első játékos 50-ediknek érkezik, az algoritmus azonnal „rájön", hogy az 50-edik játékost kell felcserélni bárki más random játékossal, és ezzel korrelálatlan mintát tud generálni, amit tovább javítani már nem képes (és alkalmasint, ezzel a módszerrel, nem is lehet). Ilyenkor a Shapley-érték becslésének hibája meg fog egyezni $2m_2$ elemű független mintáéval. A jobb oldali ábrán ugyanez látható a feszítőfa-játékra. Itt a korreláció m_1 függvényében folyamatosan csökken, és egyre kevésbé szór. Az algoritmus elég nagy (néhány ezer elemű) minta esetén megtanulja, hogy a legjobb t transzformáció a sorrend tükrözése, amely -98,5%-os korrelációt ereményez. A függvény végtelenben - úgy tűnik - egy valószínűséggel ehhez az értékhez tart. Ezzel nagyjából a negyedére csökkenthető a szórás a független mintához képest.

Korrelációval kapcsolatos megfigyelések

A másik hat játékra a 4.3. ábrán bemutatotthoz hasonló eredmények adódnak. Ezek megfigyelése alapján, bizonyítani ugyan nem tudjuk, de úgy tűnik, hogy a következő tulajdonságok teljesülnek:

- A korreláció m_1 függvényében előbb-utóbb stabilizálódik. Kis m_1-re véletlenszerű eredményt ad, de elég nagy m_1-re a szórása eltűnik, így értelmes a korrelációról, mint m_1 függvényéről beszélni.
- Elég nagy m_1 esetén a korreláció viszonylag stabil, monoton fogyó függvény, ebből kifolyólag a végtelenben létezik $\lim_{m_1 \to \infty} \rho(m_1)$ határértéke.
- A korreláció az összes vizsgált példán előbb-utóbb megközelíti a nullát. Ez nagyon fontos és pozitív eredmény, azt jelenti, hogy az algoritmus asszimptotikusan nem adhat rosszabb eredményt, mint a független minta, vagyis elég nagy minta esetén, az alkalmazásával akkor sem követünk el nagy hibát ha javítani nem tudunk.

Hogy mely játékok esetén mit jelen az „elég nagy", arra nehéz konkrétumot mondani, de elég robusztusnak tűnik a megfigyelés, hogy n-ben lineáris mennyiségű információ, $m_1 = c \cdot n$ már elegendő. Ez elméletileg is megalapozható észrevétel. Az a-adik és b-edik játékos felcserélésével az algorimus semmiféle információhoz nem jut, ha a generált mintában nincs olyan megfigyelés ahol az i-játékos az a-adik és b-edik között érkezik, mert akkor e két játékos felcserélésének nincs hatása az i határ-hozzájárulására. Minimum követelményként tehát ésszerűnek tűnik, hogy a generált mintában $\forall\ j \in \{1, 2, \ldots, n\}$-re legyen olyan sorrend, ahol i éppen j-ediknek érkezik.



4.4. ábra. Ergodikus és független minta szórásának aránya az elemszám függvényében

m_1 darab véletlen sorrend esetén annak a valószínűsége, hogy ez ne így legyen

$$\left(1-\frac{1}{n}\right)^{m_1} = \left(1-\frac{1}{n}\right)^{c\cdot n} \approx e^{-c}.$$

Ez $c = 5$ esetén már 1% alatt van, $c = 10$ esetén pedig elhanyagolhatóan kicsi, és a vizsgált példák alapján úgy tűnik, hogy ez tényleg elég is.

Becslési hiba a mintaelemszám függvényében

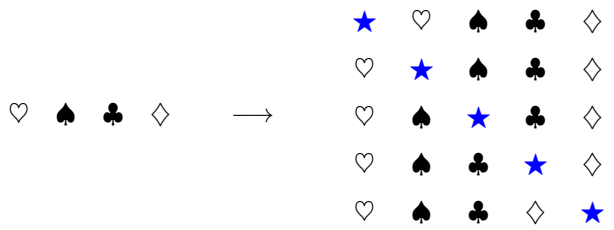
A 4.4. ábrán az m függvényében láthatjuk a σ_E/σ_R mennyiséget, ami az algoritmus „jóságának" legfőbb mércéje: hányad részére csökkenti a becslés szórását. A bal oldali grafikon tartozik a szimmetrikus szavazási játékhoz. A folytonos piros és a szaggatott narancssárga görbe két lehetséges esetet ír le, $m_1 = 50$-re. Az ábrán `A`-val jelölt pont a 4.3. táblázat hetedik sorának felel meg, itt az algoritmus az első lépésben 50 mintaelemből még nem képes megfelelő transzformációt találni, és simán megduplázza a mintát, ami tökéletes pozitív korrelációt jelent: a görbe a végtelenben $\sqrt{2}$-höz tart (vízszintes piros szaggatott vonal), azaz legjobb esetben kb. 41%-al növeli a becslési hibát. A narancssárga szaggatott vonalon lévő `D` pont a 4.3. táblázat nyolcadik sorának felel meg. Itt az algoritmus már $m_1 = 50$-re megtanulta azt a t transzformációt, mely az 50-ediknek érkező játékost felcseréli valamelyik másikkal, és két korrelálatlan mintát generál, ez a görbe végtelenben 1-hez tart, azaz asszimptotikusan ugyanakkora becslési hibával dolgozik, mint az egyszerű mintavételezés. A piros és a narancssárga esetek közül véletlenszerűen következik be valamelyik. A kék görbe az $m_1 = 1000$ paraméterhez tartozik, amikor az algoritmus már szinte biztosan a korrelálatlan mintát generálja, ezért ez a görbe is 1-hez tart végtelenben, de sokkal lassabban, mivel az első lépésben több számítási kapacitást használt fel. Kétmillió körüli m esetén még lényegesen rosszabb eredményt ad, az `A` pont helyett a `B` pontot,

de tíz millióra már jobbat (bár rosszabbat, mint a `D` pont, de azzal csak szerencsénk volt). A jobb oldali grafikon a feszítőfa-játék eredményeit mutatja. Itt az algoritmus jól működik, már $m_1 = 100$-ra erős negatív korrelációt tud elérni és már ötszázezer elemű minta esetén javít az egyszerű mintavételezéshez képest (a görbe az zöld vonal által jelzett 1 szint alá megy). Az `A`, `B` és `C` pontok a táblázat 21-23 sorai. Ha m_1 értékét följebb állítjuk 1000-re akkor még erősebb negatív korrelációt képes elérni az algoritmus, azonban ennek az ára több számítási kapacitás, ezért kis elemszám esetén (`D` pont) nagyon rossz eredményt ad, rosszabbat, mint $m_1 = 100$-ra, sőt még rosszabbat, mint az egyszerű mintavételezés (ez jól illusztrálja az m_1 és m_2 közötti nemtriviális trade-off-ot). Ha növeljük a minta elemszámát, mondjuk négy millióra akkor viszont már megéri ebből több kapacitást az első lépésben felhasználni egy jobb t transzformáció megkeresésére, és a `C` pont helyett az `E` pontba kerülni.

4.5. Összefoglaló

A statisztika módszertanából „lopott” variancia-redukció Shapley-értékre való alkalmazása nem a legegzaktabb tudomány. A hatásfoka nagyon nehezen megjósolható, általában csak olyan paraméterek ismeretében számszerűsíthető, amelyeket nem feltétlenül könnyebb kiszámolni, mint magát a Shapley-értéket, játékról játékra változ(hat)nak, és egyelőre nem látunk bennük mintákat. Annak eldöntése, hogy egy konkrét módszer egy adott játékra (vagy játékosztályra) működik-e, egy egyszerűbb probléma visszavezetése egy nehezebbre.[30]

Ennek fényében térjünk vissza Mann és Shapley hatvan évvel ezelőtti módszeréhez, és elemezzük ki alaposabban:



A módszer lényege, hogy az $i \in N$ játékos Shapley-értékének becsléséhez generálunk egy véletlen sorrendet az összes többi játékosból, majd ebbe a sorba minden lehetséges helyre betesszük az i játékost, így n darab különböző sorrendet kapva. Ezekre kiszámoljuk az i játékos határhozzájárulását, és tekintjük ezek átlagát. Mindezt m/n-szer megismételve összesen m darab véletlen sorrendből becsüljük a Shapley-értéket, melyek azonban nem függetlenek, így remélhetőleg kisebb szórású becslést kapunk mint egy m elemű független mintából. Mivel Mann és

[30] Miért lenne könnyebb megmondani, hogy mennyi California határ-hozzájárulásának szórása azon feltétel mellett, hogy tizenötödiknek csatlakozik a többiekhez, mint magát a Shapley-értéket?

Shapley (1960) kifejezetten az *Electoral College* Shapley-értékét szerette volna kiszámolni, egyáltalán nem foglalkozik azzal, hogy a módszer működik-e bármely más játékosztályra, és első ránézésre ez egyáltalán nem is nyilvánvaló. Az n darab sorrendhez tartozó határ-hozzájárulások egy része még a szavazási játékok esetén is pozitívan korrelál, ami pedig nem csökkenti, hanem növeli a becslési hibát.

Legyen $i \in N$ egy játékos és tekintsük a játékosok N halmazának összes lehetséges sorrendjeiből álló $\mathcal{O}(N)$ halmazon a következő relációt:

$$o_1 \sim o_2 \overset{\text{def}}{\iff} \quad \forall i \neq i_1, i_2 \text{ -re } \; o_1(i_1) \leq o_1(i_2) \Leftrightarrow o_2(i_1) \leq o_2(i_2).$$

Két sorrend tehát ekvivalens, ha az i-től különböző játékosok ugyanabban a sorrendben vannak. Nyilvánvaló, hogy ez ekvivalencia-reláció, melynek $(n-1)!$ darab n elemű osztálya van. Jelölje $\mathcal{A}$ a reláció ekvivalencia-osztályai által generált σ-algebrát. A Shapley-Mann algoritmus a $\mathbb{E}(v_i' \mid \mathcal{A})$ feltételes eloszlásból generál független mintát és ennek várható értékére ad becslést. A feltételes várható érték $\mathcal{L}_2$-ben vetítés, a szórásnégyzete nem nagyobb, mint a változó szórásnégyzete, várható értéke pedig maga a várható érték, így a módszer torzítatlan, konzisztens, és a becslési hibája legfeljebb akkora, mint a független mintáé, ezért, ha ezzel a módszerrel generálunk $n \cdot \frac{m}{n}$ véletelen sorrendet, abból jobb becslést kapunk a Shapley-értékre, mint egy ugyanennyi m elemű független mintából. Csakhogy két Monte-Carlo algorimust nem a generált minta mérete, hanem a futási idejük alapján kell összehasonlítani. Nézzük meg részletesen, hogyan számolja ki a szavazási játék esetén a sima Monte-Carlo eljárás az i játékos határ-hozzájárulását a játékosok egy véletlen sorrendjéhez.

1. Legyen $k = 1$ és legyen $s = 0$.
2. Tekintsük a k-adik játékost, aki legyen i_1.
   - Ha $i = i_1$ és $s < q - w_i$ akkor `return` 0
   - Ha $i = i_1$ és $s \geq q - w_i$ akkor `return` 1
3. $s = s + w_{i_1}$, $k = k + 1$.
4. Ha $s \geq q$ akkor return 0 egyébként `GoTo (2).`

Érkezési sorrendben elkezdjük összeadogatni a játékosok súlyait és ezek kummulált értékét egy változóba mentjük. Ha megtaláljuk az i játékost, és a játékosok kummulált szavazóereje vele együtt is kisebb, mint a kvóta, vagy nem találtuk meg, de nélküle is nagyobb, akkor visszatérünk nullával. Ha az i játékost pont akkor találjuk meg, amikor a játékosok kummulált szavazóereje még kisebb, mint a kvóta, de vele együtt már nagyobb, akkor visszatérünk 1-gyel.

Mit csinál ugyanezzel a sorozattal a Shapely-Mann algoritmus?

1. Legyen $k = 1$, $s = 0$ és $cntr = 0$.
2. Ha $w_i > q$ akkor $cntr = cntr + 1$
3. Tekintsük a k-adik játékost, aki legyen i_1.
4. $s = s + w_{i_1}$
   - Ha $s \geq q$ akkor return $cntr/n$
   - Ha $s \geq q - w_i$ akkor $cntr = cntr + 1$.
5. $k = k + 1$, `GoTo (3)`

Létrehozunk egy számlálót, melyet mindig megnövelünk 1-gyel, ha az i játékos az aktuálisan vizsgált játékos után, vagy elsőnek érkezve, éppen megfordítaná a szavazás eredményét. Érkezési sorrendben elkezdjük összeadogatni a játékosok súlyait és ezek kummulált összege $q - w_i$ és q között van, akkor növeljük a számlálót. Ha a játékosok kummulált szavazóereje eléri a kvótát kilépünk.

A két algoritmus lényegében ugyanazt csinálja, csak a második nyilvántart még +1 számlálót, tehát minden iterációban 1-gyel több műveletet végez, mely lényegében elhanyagolható. Ha nagyon precízek akarunk lenni, akkor a különbség a ciklus hosszában van. A második algoritmus mindenképpen addig iterál a játékosokon, amíg azok kummulált szavazóereje el nem éri a kvótát, míg az első hamarabb is kilép, ha megtalálja az i játékost. Mivel a játékosok véletlenszerűen állnak sorba egyenletes eloszlással, az első algoritmus nagy átlagban kb. fele annyi lépés után kilép, mint a második és minden lépésben kicsit kevesebb műveletet végez, tehát legyen mondjuk, négyszer gyorsabb. Ha tehát a futási idejük alapján akarjuk őket úgy összehasonlítani, hogy a Shapley-Mann módszer semmiképpen se legyen előnyben, akkor mondjuk negyedannyi véletlen sorrendet kell vele generáltatni, mint a sima Monte-Carlo algoritmussal. Ennek eredményeként azonban az utóbbi minden sorrendhez egy darab, az előbbi viszont n darab határ-hozzájárulást tud kiszámolni. A Shapley-Mann algoritmus becslési hibája tehát nagyjából $2/\sqrt{n}$-szerese az egyszerű független mintavételezésnek[31], vagyis annál többet javít, minél nagyobb a játék, még akkor is, ha eltekintünk a feltételes várható érték eredetitől kisebb szórásától (amit ugyanúgy nem tudunk megjósolni, ahogy az összes többi módszernél sem).

Az *Electoral College*-ra (ez egy 51 játékosból álló játék) futtatott numerikus szimuláció eredményeként azt kaptuk, hogy a Shapley-Mann algoritmus becslési hibája negyedakkora minta generálásával 23%-a a független mintáénak. Ilyen eredményt a legújabb módszerek (Maleki et al. (2013), Castro et al. (2017), Illés és Kerényi (2019)) csak a legjobb esetekben és igen ritkán produkálnak, és ez több játékos esetén feltehetően még durvább.

[31] Ennyi lenne, ha független megfigyeléseket generálna, így nem tudjuk pontosan.

Kérdés, hogy a Shapley-Mann módszer működik-e más játékosztályokra. Az, hogy lényegében ugyanannyi művelettel ki tudtunk számolni n darab határ-hozzájárlást, mint amennyi művelettel egy darabot ki lehet számolni, azon múlott, hogy minden $i \in N$ játékos és $S \subseteq N$ koalíció esetén az $S \cup \{i\}$ koalíció $v(S \cup \{i\})$ értéke konstans időben, vagy elhanyagolhatóan kevés művelettel kiszámolható volt az S koalíció $v(S)$ értékének ismeretében, mert a játékosok súlyainak kummulálását nem kellett minden lépésben előlről kezdeni, amikor egy új játékos a koalícióhoz csatlakozott. Ez a helyzet minden olyan játék esetében fennáll például, amelynek karakterisztikus függvénye megadható

$$v(S) = f\left(\sum_{i \in S} a_i\right)$$

alakban, ahol az a_i súlyok a játékosokhoz rendelt valamilyen paraméterek, f pedig egy „gyorsan" kiszámolható függvény[32]. Ezesetben ugyanis bármely i játékosra a játékosok egy tetszőleges sorrendjéhez egyetlen ciklussal ki lehet számolni az i játékosnak a sorozat kezdőszeleteiből álló összes koalícióhoz való határ-hozzájárulását, pontosan úgyanúgy, ahogy a szavazási játékok esetében. A feszítőfa-játék például nem ilyen. A minimális költségű feszítőfát nemtriviális módon változtatja meg egy új csúcs hozzávétele a gráfhoz, mert olcsó élekkel összeköthet olyan komponenseket, amelyeket nélküle csak drágábban lehetett összekötni. Egy új játékos csatlakozása esetén a koalíció értékét „előlről kell kezdeni" kiszámolni. A legtöbb játékunk (csődjáték, szavazási játék, repülőtér-játék, tartozásos játékok) azonban ilyen. Ezekre a Shapley-Mann algoritmus valószínűleg kiválóan működik, meglehet[33], annyira kiválóan, hogy az elmúlt hatvan évben nem sikerült jobbat találni.

[32] Ilyen játékokról szól a következő fejezet, ott pontosabban is definiáljuk majd.

[33] Ez egyelőre egy sejtés, további kutatást igényelne eldönteni.

5. fejezet

Egzakt, pszeudopolinomiális algoritmusok

A pszeudopolinomiális algoritmusokat a 2.5 fejezetben vezettük be. A fogalomnak csak olyan problémák esetén van értelme, ahol az input számokból vagy számsorozatokból áll. A TU-játékok tipikusan ilyenek, ezért van értelme a Shapley-érték kiszámítására olyan algoritmust keresni, mely ugyan nem polinomiális az input méretének függvényében, de a futási ideje, a brute force módszerrel ellentétben, a játékosok számával nem növekszik exponenciálisan.

Noha ezt a terminológiát sohasem használták, Mann és Shapley (1962) is egy pszeudopolinomiális algoritmust ad szavazási játékok Shapley-értékének kiszámítására. Az algoritmus a (szokásos értelemben) polinomiális olyan rész-játékosztályokon, ahol a súlyok a játékosok számának egy polinomjával korlátozhatóak. A fejezet hátralévő részében a célunk ezt az eredményt Illés (2022) alapján kiterjeszteni a szavazásos játékoknál bővebb játékosztályra. Ehhez hasznos lesz (és ettől függetlenül is érdekes) a Shapley-értéket egy kevésbé használatos, alakban felírni, melyet Owen (1972) multilineáris kiterjesztésének egyik alkalmazásaként Leech (2003) használ szintén szavazási játékok esetére.

5.1. Multilineáris kiterjesztés

Jelölje pozitív $a, b \in \mathbb{N}$ egészekre $\beta(a, b)$ az Euler-féle béta függvényt, azaz:

$$\beta(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\, dx.$$

Könnyen látható, hogy parciális integrálással:

$$\int_0^1 x^{a-1}(1-x)^{b-1}\,dx = \underbrace{\left[\frac{x^a}{a}(1-x)^{b-1}\right]_{x=0}^{x=1}}_{0} - \int_0^1 -\frac{x^a}{a}(b-1)(1-x)^{b-2}\,dx$$

azaz:

$$\beta(a,b) = \frac{b-1}{a}\beta(a+1,b-1)$$

amiből b szerinti teljes indukcióval könnyen adódik a közismert

$$\beta(a,b) = \frac{(a-1)!\cdot(b-1)!}{(a+b-1)!}$$

formula. Az első fejezet (1.2) képlete szerint:

$$Sh_i = \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!\cdot(n-|S|-1)!}{n!}\cdot v_i'(S)$$

ami $a = |S|+1$, $b = n-|S|$ jelöléssel:

$$Sh_i = \sum_{S\subseteq N\setminus\{i\}} \frac{(a-1)!\cdot(b-1)!}{(a+b-1)!}\cdot v_i'(S) = \sum_{S\subseteq N\setminus\{i\}} \int_0^1 x^{a-1}(1-x)^{b-1}\cdot v_i'(S)\,dx =$$

$$\sum_{S\subseteq N\setminus\{i\}} \int_0^1 x^{|S|}(1-x)^{n-|S|-1}\cdot v_i'(S)\,dx = \int_0^1 \sum_{S\subseteq N\setminus\{i\}} x^{|S|}(1-x)^{n-|S|-1}\cdot v_i'(S)\,dx.$$

Az $x^{|S|}(1-x)^{n-|S|-1}$ kifejezés annak a valószínűsége, hogy ha minden $N\setminus\{i\}$ -beli játékos egymástól függetlenül x valószínűséggel dönt a kooperáció mellett, akkor a kooperáló játékosok halmaza éppen S. Mivel ezt minden lehetséges $S\subseteq N\setminus\{i\}$ koalícióra kiszámoljuk és összeadjuk, az integrál mögötti súlyozott összeg is egy várható értékként értelmezhető. Emellett egy függvény integrálja 0-tól 1-ig (1-el osztva) is a függvény átlagértékének tekinthető. Ezzel a Shapley-érték egy új interpretációját nyerjük, mint a határ-hozzájárulás várható értékét.

$$Sh(i) = \int_0^1 \mathbb{E}_x v_i'(S)\,dx \tag{5.1}$$

ahol az $\mathbb{E}_x$ jelölést úgy kell értelmezni, hogy az i játékoson kívül az összes többi játékos x valószínűséggel egymástól függetlenül (szimultán) dönt, hogy kooperál-e és tekintjük az i játékos határ-hozzájárulását a kooperáló játékosokból álló véletlen koalícióhoz. A Shapley-érték ennek a várható értéknek az integrálja az x valószínűség szerint, azaz ennek a várható értéknek a várható értéke, az x-et is egyenletes eloszlással választjuk 0 és 1 között.

5.2. Lineárisan reprezentálható játékok

A Shapley-érték pszeudopolinomiális algoritmussal az alábbi játékok esetében számolható ki:

5.2.1. Definíció. *Egy (N, v) TU-játékra azt mondjuk, hogy lineárisan reprezentálható, ha a játékosokhoz rendelhetőek $a_j : j \in N$ súlyok és létezik $f : \mathbb{N} \to \mathbb{N}$ függvény, hogy minden $S \subseteq N$ koalíció értéke előáll $v(S) = f\left(\sum_{i \in S} a_i\right)$ alakban. Ezesetben a fenti f függvényt és az a_j számokat a játék egy lineáris reprezentációjának nevezzük. Ha egy játéknak adott egy lineáris reprezentációja, akkor azt mondjuk, hogy a játék ez által lineárisan reprezentált.*

Megjegyzés. Egy játéknak lehet több különböző lineáris reprezentációja is.

A lineárisan reprezentálhatóság látszólag azt jelenti, hogy a játék megadható a játékosok számával megegyező számú paraméterrel, a következő tétel szerint azonban ez nem a helyes megközelítés.

5.2.1. Tétel. *Minden TU-játék lineárisan reprezentálható.*

Bizonyítás. Számozzuk be tetszőleges sorrendben a játékosokat 0-tól $n-1$-ig és rendeljük a j-edik játékoshoz a 2^j számot, az f függvény pedig minden $1 \leq k \leq 2^n$ számra legyen annak az egyértelműen létező S koalíciónak a $v(S)$ értéke, melyre $k = \sum_{i \in S} a_i$. Ilyen S koalíció létezik és egyértelmű, mert minden szám egyértelműen áll elő kettőhatványok összegeként, és könnyen megtalálható, mert k kettes számrendszerbeli számjegyei épp a neki megfelelő koalíció indikátorvektora. □

Triviálisnak tűnik tehát, hogy minden játék lineárisan reprezentálható, azonban vegyük észre, hogy a látszólag konstruktív bizonyítás ellenére, ez egy egzisztenciatétel. Bár a játékosokhoz rendelt súlyok exponenciálisan nagyok, ami – mint látni fogjuk – a Shapley-érték kiszámolásában nem segít, a fő probléma nem ez, hanem az, hogy az f függvényről nem tudjuk, hogyan kell kiszámolni. A függvény valójában egy orákulum, aki ismeri a játékot, azaz minden koalíció értékét hozzá tudja rendelni egy egész számhoz annak bináris reprezentációja alapján. Egy játék ilyetén leírása nem tesz eleget a polinomiális reprezentáció 3.1.1. definíciójának, mert a koalíciók értékét kiszámoló Turing-gépnek át kell adni a koalíciók értékeinek teljes listáját is, ami exponenciális mennyiségű információ. Egy játék lineáris reprezentációja akkor hasznos, ha a benne szereplő súlyok n-ben polinomiálisak, az f függvény pedig a játékosok számában polinomiális mennyiségű információ alapján kiszámolható, ahogy az alábbi példák mutatják:

- A többségi szavazásos játékok lineárisan reprezentálhatóak $a_i = w_i$ súlyokkal és az

$$f(x) = \begin{cases} 1 & \text{ha } x \geq q, \\ 0 & \text{ha } x < q. \end{cases}$$

  függvénnyel.

- Könnyen látható, hogy a csődjátékok is lineárisan reprezentálhatóak $a_i = l_i$ súlyokkal és az $f(x) = \max\{0, A - (L - x)\}$ függvénnyel, ahol $L = \sum l_j$.

- A 3.3.1. alfejezetben bemutatott tartozásos játékok is lineárisan reprezentálhatóak, de ez nem teljesen triviális a karakterisztikus függvényben lévő esetszétválasztás miatt. A trükk az, hogy a fizetésképtelen céget reprezentáló 0-s játékoshoz egy elég nagy súlyt rendelünk ahhoz, hogy a súlyok összege alapján szét lehessen választani a két esetet (azaz meg lehessen mondani, hogy a cég tagja-e az adott koalíciónak vagy sem). Legyen tehát $L = \sum l_i + 1$, $a_0 = L$, $a_i = l_i$ minden $i \geq 1$-re és legyen

$$f(x) = \begin{cases} \min\{A, x - L\} & \text{ha } x \geq L \\ \max\{0, A - (L - 1 - x)\} & \text{ha } x < L. \end{cases}$$

  Vegyük észre, hogy az egyetlen új bevezetett L súly n-ben még mindig polinomiális, ha az eredeti l_i-k azok voltak.

5.3. Lineárisan reprezentált játékok Shapley-értéke

A következőkben bemutatásra kerülő algoritmus több ismert eredmény közös általánosítása, többek között a következő három, egymástól látszólag távoleső probléma megoldásának:

- Mann és Shapley (1962) által szavazási játékokra javasolt algoritmus, Cantor ötlete alapján, a generátorfüggvény rekurzív kiszámításával.
- Polinomiális algoritmus a 3.1. fejezetben bemutatott repülőtér-játékok Shapley-értékére.
- Aziz (2013) által javasolt algoritmus csődjátékok Shapley-értékének kiszámítására dinamikus programozással.

Megjegyzés. Noha beláttuk, hogy minden játék lineárisan reprezentálható, mivel a reprezentáció általában (legrosszabb esetben) exponenciális, a bemutatásra kerülő algoritmus nem segít olyan játékok Shapley-értékének kiszámításában például, ahol a játék egy gráffal, (a játékosok számában négyzetes méretű paraméterhalmazzal) van adva, még akkor sem, ha az adott játékra a Shapley-érték egyébként polinom-időben számolható. Ilyen esetre példa a 3.2. fejezetben megismert költségallokációs játék.

5.3.1. Lineárisan reprezentálható játékok főtétele

Most ismertetjük a fejezet fő eredményét.

5.3.1. Tétel. *Legyen a $G = (N, v)$ játék egy lineáris reprezentációja $\big(f, \{a_j : j \in N\}\big)$. Tegyük fel, hogy minden $a_j \leq 2^j$, és az f függvény lineáris időben és tárban kiszámolható. Ekkor tetszőleges $i \in N$ játékos Shapley-értéke a G játékban kiszámítható $O(n^3K)$ időben és $O(n^2K)$ tárban, ahol $K = \sum_{j \neq i} a_j$.*

Megjegyzés. A súlyok nagyságára feltett exponenciális korlát az 5.2.1. tétel alapján mindig elérhető, az f-re tett feltételek pedig technikai jellegűek (és az eddigi példáknál fennállnak), a tétel, értelemszerű módosításokkal, ezek nélkül is érvényes.

Bizonyítás. Az algoritmus a Shapley-értéket az

$$Sh(i) = \int_0^1 \mathbb{E}_x v_i'$$

(5.1) egyenlet felhasználásával fogja kiszámolni. Legyen $S \subseteq N$ egy tetszőleges, az i játékost nem tartalmazó koalíció. A lineáris reprezentáció szerint $v(S) = f\left(\sum_{j\in S} a_j\right)$, amiből:

$$v_i'(S) = f\left(\sum_{j\in S} a_j + a_i\right) - f\left(\sum_{j\in S} a_j\right)$$

Legyen most $\mathcal{S}_x \subseteq N \setminus \{i\}$ az a véletlen koalíció, melyet az i nélküli $n-1$ játékos alkot, midőn egymástól függetlenül x valószínűséggel a kooperációról döntenek. Ezzel a jelöléssel

$$\mathbb{E}_x v_i' = \sum_{S\subseteq N\setminus\{i\}} \mathbb{P}\left(\mathcal{S}_x = S\right) \cdot \left(f\left(\sum_{j\in S} a_j + a_i\right) - f\left(\sum_{j\in S} a_j\right)\right) =$$

Legyen $K = \sum_{j\in N\setminus\{i\}} a_j$ és csoportosítsuk az összeg tagjait az S-beli súlyok összege szerint:

$$= \sum_{k=0}^{K} \sum_{\substack{S\subseteq N\setminus\{i\} \\ \sum_{j\in S} a_j = k}} \mathbb{P}\left(\mathcal{S}_x = S\right) \cdot \Big(f\left(k + a_i\right) - f\left(k\right)\Big) =$$

$$= \sum_{k=0}^{K} \Big(f\left(k + a_i\right) - f\left(k\right)\Big) \cdot \sum_{\substack{S\subseteq N\setminus\{i\} \\ \sum_{j\in S} a_j = k}} \mathbb{P}\left(\mathcal{S}_x = S\right) = \sum_{k=0}^{K} \Big(f\left(k + a_i\right) - f\left(k\right)\Big) \cdot \mathbb{P}(\mathcal{G}_x = k),$$

ahol $\mathcal{G}_x$ jelöli a $\sum_{j\in\mathcal{S}_x} a_j$ valószínűségi változót, azaz, a játékosok által kialakított véletlen koalíció összsúlyát, midőn egymástól függetlenül x valószínűséggel a kooperációról döntenek. A $\mathbb{P}(\mathcal{G}_x = k)$ valószínűségek az x változó egész együtthatós $(n-1)$-ed fokú polinomjai, melyeket rekurzíve ki tudunk számolni, a következőképpen:

- Tekintsük az a_i-n kívüli súlyok (vagyis az i-től különböző játékosok) egy tetszőleges, de előre rögzített sorrendjét, és legyen $P[j,k]$ annak a valószínűsége, hogy az első j játékos egy k összsúlyú koalíciót alkot ($0 \le j \le n-1$ és $0 \le k \le K$)[34]. Ez minden j-re és k-ra az x változó egy j-edfokú polinomja. Nyilván $j = 0$-ra $P[0,0] = 1$ és $P[0,k] = 0$ $k \ge 1$-re.

[34] Már itt jegyezzük meg, hogy a $P[j,k]$ mátrix a játékosok egy előre rögzített, önkényes sorrendjétől függ, amitől a Shapley-érték természetesen nem függhet, és nem is fog. Erre később még visszatérünk.

- Minden $1 \leq j \leq n-1$-re, ha $P[j-1,k]$ már ismert minden $0 \leq k \leq K$-ra, akkor a teljes várható érték tétele szerint:

$$P[j,k] = \begin{cases} (1-x) \cdot P[j-1,k] & \text{ha } k < a_j \\ x \cdot P[j-1,k-a_j] + (1-x) \cdot P[j-1,k] & \text{ha } k \geq a_j \end{cases}$$

A $P[n-1,k]$ számok épp a $\mathbb{P}(\mathcal{G}_x = k)$ valószínűségek. Ezek egyszerű súlyozott összegeként kapjuk a $\mathbb{E}_x v_i' = \sum_{k=0}^{K} \Big(f(k+a_i) - f(k) \Big) \cdot \mathbb{P}(\mathcal{G}_x = k)$ kifejezést, mely szintén egy $n-1$-ed fokú polinom, így az integrálját is ki tudjuk számolni analítikusan: egyszerűen kiintegráljuk tagonként (a konstans tagot nullának tekintve), majd a primitív függvény (egy n-ed fokú polinom) együtthatóit összeadjuk. Ez egy racionális szám, a játék pontos Shapley-értéke.

Ami az algoritmus komplexitását illeti, első ránézésre úgy tűnik, hogy egy $n \times K$-as mátrixot kell kitölteni, melynek minden eleme algebrai alapműveletekkel (egyetlen lépésben) megkapható, azonban ez kicsit félrevezető, ennyire azért nem jó a helyzet. A mátrixban ugyanis egyrészt polinomokat tárolunk, tehát praktikusan n darab együtthatót, másrészt, ezek az együtthatók n-ben exponenciálisan nagyok lehetnek, amit az olyan egyszerűnek látszó műveleteknél, mint az összeadás vagy szorzás, is figyelembe kell venni. A mátrix első sorában csak olyan (konstans) polinomok vannak, melyek mindegyik együtthatója 0 vagy 1. Minden sor legfeljebb három, előző sorban lévő polinom összege (nem csak kettő, hanem három, mert az $(1-x)$-es tag eleve két polinom összege, és ehhez jön még az x-es tag). így j szerinti indukcióval $P[j,k]$ mindegyik együtthatója legfeljebb 3^j ami legfeljebb 3^n, ami egy legfeljebb $O(n)$ bites szám. Legfeljebb n darab legfeljebb $O(n)$ bites számot tárolni legfeljebb $O(n^2)$ tárhely, és két ilyet összeadni legfeljebb $O(n^2)$ művelet. A $P[j,k]$ mátrix tehát kiszámolható $O(n^3K)$ futási idővel. Az algoritmus során szükségtelen a teljes mátrixot tárolni, elegendő mindig a két aktuális sort tárolni, ami csak $O(n^2K)$ memóriát igényel. Könnyen látható, hogy az f-re tett egyszerűsítő feltevések miatt, a $P[n-1,k]$ polinomok előállítása a leginkább számításigényes művelet, az algoritmus többi lépése (az f függvény kiértékelése K pontban, a polinomok összesúlyozása, és integrálása) elvégezhető $O(n^2K)$ lépésben, a teljes algoritmus futási ideje tehát $O(n^3K)$. □

5.3.1. Példa. Illusztrációként tekintsük azt a csődjátékot, amelyben négy hitelező követelései rendre $l_1 = 2$, $l_2 = 3$, $l_3 = 5$, $l_4 = 7$, az eszközök értéke $A = 9$, és számítsuk ki a negyedik játékos Shapley-értékét. A $P[j,k]$ mátrixot a 5.1. táblázat tartalmazza. A mátrix első sora azt fejezi ki, hogy az üres halmaz 1 valószínűséggel egy 0 súlyú koalíciót alkot, a többi sor a kezdet kezdetén nullákkal van feltöltve. Ezután minden sortból úgy kapjuk az alatta lévőt, hogy minden elem $1-x$-szeresét hozzáadjuk az alatta lévő elemhez, illetve x-szeresét hozzáadjuk a tőlet l_j-vel jobbra lévő oszlop eggyel lejjebb lévő eleméhez, ahol l_j a nagyság szerinti (vagy bármely tetszőleges előre rögzített sorrend szerinti) j-edik súly. Fejéről a talpára állítva, a

mátrix minden eleme a közvetlenül fölötte lévő polinom $(1-x)$-szeresének és az l_j-vel balra, a fölötte lévő sorban lévő (ha van ilyen) polinom x-szeresének az összege. A mátrixban lévő polinomokat ki kell fejteni monomok összegére, ahogy azt a 5.2. táblázatban láthatjuk, hogy a végén könnyen ki tudjuk integrálni.[35] A táblázat utolsó sorában k összköveteléssel rendelkező koalíció kialakulásának valószínűségei vannak $k = 0, 1, \ldots, 10$-re. Ha egy S koalíció összes követelése k, akkor a komplementer koalícióé $2+3+5+7-k$. Az eszközök értékének (ami $A = 9$) ezt meghaladó része azaz $(A-(17-k))^+ = (k-8)^+$ jut az S koalíciónak. Ha a negyedik játékos egy koalícióhoz csatlakozik, akkor annak összsúlyát, azaz a követelések összértékét $l_4 = 7$-el növeli, így az $(k-1)^+$ lesz. A játékos határ-hozzájárulása e két érték különbsége, melyet feltüntettük a 5.2. táblázat alatt. Ezekkel a számokkal kell összesúlyozni a fölötte lévő sorban lévő polinomokat. A negyedik játékos várható határ-hozzájárulása így:

$$x^3-2x^2+x+2(x^3-2x^2+x)+4(-x^2+x)+6(-x^3+x^2)+7(-x^3+x^2)+7x^3 = -3x^3+3x^2+7x$$

A Shapley-érték ennek x szerinti integrálja 0-tól 1-ig, azaz a primitív függvény együtthatóinak összege: $-\frac{3}{4}+1+\frac{7}{2} = \frac{15}{4}$.

[35] Elméletileg megpróbálhatnánk a mátrixban lévő polinomokat $x^u(1-x)^v$ alakú tagok összegeiként is tárolni. Az ilyen alakú polinomokat is ki tudjuk integrálni, hiszen ez épp az 5.1 fejezetben már tárgyalt β függvény, de nem világos, hogy ez érdemben gyorsítaná-e az algoritmust.

	0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	0	0	0	0	0	0	0	0
$l_1 = 2$	$1-x$	0	x	0	0	0	0	0	0	0	0
$l_2 = 3$	$(1-x)^2$	0	$x(1-x)$	$x(1-x)$	0	x^2	0	0	0	0	0
$l_3 = 5$	$(1-x)^3$	0	$x(1-x)^2$	$x(1-x)^2$	0	$x^2(1-x)+x(1-x)^2$	0	$x^2(1-x)$	$x^2(1-x)$	0	x^3

5.1. táblázat. A 5.3.1. példa polinomjainak rekurzíve előállítása

	0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	0	0	0	0	0	0	0	0
$l_1 = 2$	$-x+1$	0	x	0	0	0	0	0	0	0	0
$l_2 = 3$	x^2-2x+1	0	$-x^2+x$	$-x^2+x$	0	x^2	0	0	0	0	0
$l_3 = 5$	$-x^3+3x^2-3x+1$	0	x^3-2x^2+x	x^3-2x^2+x	0	$-x^2+x$	0	$-x^3+x^2$	$-x^3+x^2$	0	x^3

	0	0	1	2	3	4	5	6	7	7	7

5.2. táblázat. A 5.3.1. példa polinomjainak felbontása

5.3.2. Shapley-érték az összes játékosra

Ha egy játékban egy tetszőleges játékos Shapley-értékét ki tudjuk számolni, akkor nyilván ezt n-szer alkalmazva megkaphatjuk az összes játékos Shapley-értékeiből álló vektort, így az 5.3.1. tétel triviális következménye az alábbi

5.3.1. Következmény. *A 5.3.1. tétel feltételei mellett, tetszőleges lineárisan reprezentálható játék Shapley-értéke előállítható $O(n^4K)$ időben, ahol $K = \sum a_j$.*

Gyanús azonban, hogy ha a $P[j,k]$ mátrixot már egyszer előállítottuk, akkor talán felesleges a teljes számolást játékosonként előlről kezdeni. A $P[j,k]$ mátrix definíció szerint függ az i játékostól, akinek a Shapley-értékét ki akarjuk számolni, mert éppen őt kell kihagyni, és függ az összes többi játékos egy előre rögzített, önkényes sorrendjétől. Az i játékos azonban, az ezen önkényes sorrend szerinti utolsó játékosra könnyen kicserélhető. Például a 5.2. táblázatban látható hogy, ha az $l_4 = 7$ súlyú játékos helyett a utolsó előtti $l_3 = 5$ súlyú játékos Shapley-értékét akarnánk kiszámolni, akkor elegendő lenne az utolsó sort újraszámolni, a mátrix első két sora változatlan. Ez persze (látszólag) nem segít, ha például az (eredeti sorrend szerinti) első játékos Shapley-értéke érdekel minket, mert úgy tűnik, hogy ahhoz egész az első sorig vissza kellene bontani a mátrixot, de érdemes rajta elgondolkodni, hátha mégsem. A továbbiakban azt fogjuk belátni, hogy valójában elegendő a mátrix összes játékoshoz tartozó utolsó sorát ismerni, abból az összes játékos Shapley-értéke gyorsan kiszámolható. Ez, röviden össszefoglalva, azon az észrevételen alapul, hogy a mátrix utolsó sora nem függ a játékosok (azaz a súlyok) sorrendjétől, és a játékosok (súlyok) tetszőleges sorrendjéhez a mátrix visszafelé is előállítható, azaz bármely sorából visszakapható ez előző. Nézzük mindezt részletesen:

5.3.1. Lemma. *Legyen $j_1, j_2, \ldots, j_{n-1}$ az i-től különböző játékosok egy tetszőleges sorrendje, és tekintsük a 5.3.1. tétel szerinti $P[j,k]$ mátrixot. Ekkor:*

a) Ha az ℓ-edik és $\ell+1$-edik játékost felcseréljük, akkor a $P[j,k]$ mátrix ℓ-edik során kívül egyis sem változik.

* *Speciálisan: a mátrix j-edik sora nem függ az első j játékos (sem az utolsó $n-1-j$ játékos) egymás közötti sorrendjétől.*

** *Még speciálisabban: a mátrix utolsó sora független a játékosok sorrendjétől.*

b) A játékosok előre rögzített sorrendjének ismeretében a mátrix tetszőleges sorából megkapható az előző sor.

Bizonyítás.

a) A lemma a) része triviális, ha elhisszük, hogy $P[j,k]$ annak a valószínűsége, hogy az első j játékos egy k összsúlyú koalíciót alkot, ez nyilván nem függhet a játékosok sorrendjétől. Ha erről közvetlen számolással is meg akarunk győződni, akkor a $P[j,k]$-ra vonatkozó rekurziót $j=\ell+1$-re, majd $j=\ell$-re is felírva (az értelmetlen indexekhez tartozó elemeket nullának tekintve), kapjuk, hogy:

$$\begin{aligned}P[\ell+1,k] &= x\cdot P[\ell,k-a_{\ell+1}]+(1-x)\cdot P[\ell,k]= \\ &= x\cdot\Big(x\cdot P[\ell-1,k-a_{\ell+1}-a_\ell]+(1-x)\cdot P[\ell-1,k-a_{\ell+1}]\Big)+ \\ &\quad +(1-x)\cdot\Big(x\cdot P[\ell-1,k-a_\ell]+(1-x)\cdot P[\ell-1,k]\Big)= \\ &= x^2\cdot P[\ell,k-(a_{\ell+1}+a_\ell)]+x(1-x)\Big(P[\ell-1,k-a_{\ell+1}]+P[\ell-1,k-a_\ell]\Big)+(1-x)^2\cdot P[\ell-1,k]\end{aligned}$$

ami pedig a_ℓ-ben és $a_{\ell+1}$-ben kétségtelenül szimmetrikus, így a rekurzió két lépésben már érzéketlen arra, hogy ezt a két lépést milyen sorrendben hajtjuk végre.

b) A lemma b) részének bizonyításához vegyük észre, hogy, ha a mátrix sorait K-dimenziós vektoroknak tekintjük, akkor a $j-1$-edik sorból a j-edik sor egy $\chi^K(x)\to\chi^K(x)$ lineáris transzformációval kapható, ahol $\chi(x)$ az x határozatlanú egyváltozós polinomgyűrű hányadosteste. A transzformáció mátrixa diagonális, főátlójában végig az $1-x$ polinom áll, a_j-vel a főátló alatt pedig az x polinom. Mivel ez alsó háromszög-mátrix, a főátlóban nemnulla elemekkel, a determinánsa sem nulla így invertálható. Azt is könnyű látni, hogyan kapható meg a j-edik sorból a $j-1$-edik. Például a $P[j,k]$ mátrix első oszlopa mindig $(1-x)^j$, sőt általában, ha $k<a_j$ akkor $P[j,k]=P[j-1,k]\cdot(1-x)$, így a j-edik sor első a_j-1 eleme mindig osztható $(1-x)$-el (maradék nélkül) és $P[j-1,k]=P[j,k]/(1-x)$. Innentől rekurzíve balról jobbra haladva a $(j-1)$-edik sor többi eleme is kiszámolható, hiszen ha $k\geq a_j$ és minden $l<k$-ra $P[j-1,l]$ már ismert, akkor

$$P[j,k]=x\cdot P[j-1,k-a_j]+(1-x)\cdot P[j-1,k]$$

miatt $P[j,k]-x\cdot P[j-1,k-a_j]$ osztható $(1-x)$-el, és

$$P[j-1,k]=\big(P[j,k]-x\cdot P[j-1,k-a_j]\big)/(1-x)$$

Ezzel a lemma mindkét állítását beláttuk. □

Nézzük, hogyan használható ez a Shapley-értékek számolásának felgyorsítására.

5.3.2. Következmény. *Legyen a $G=(N,v)$ játék egy lineáris reprezentációja $\big(f,\{a_j:j\in N\}\big)$, ahol $a_j\leq 2^j$, és az f függvény lineáris időben és tárban kiszámolható, azaz teljesülnek a 5.3.1. tétel feltételei. Ekkor a játék Shapley-értéke n kiszámítható $O(n^3K)$ időben és $O(n^2K)$ tárban, ahol $K=\sum a_j$.*

Bizonyítás. Az 5.3.1. tétel bizonyításában bemutatott algoritmust módosítjuk az alábbiak szerint. Ahelyett, hogy először kiválasztanánk egy i játékost, akinek a Shapley-értékét tudni szeretnénk, először kiszámoljuk a $P[j,k]$ mátrixot az összes játékosra, azaz az összes a_j súly figyelembevételével. A mátrix így $(n+1) \times K$-as lesz és nyilván még mindig $O(n^3K)$ időben előállítható. Legyen most $i \in N$ egy tetszőleges játékos. A 5.3.1. lemma szerint a mátrix utolsó sorában előálló $P[n,k]$ polinomok (nevezzük ezeket a továbbiakban az egyszerűség kedvéért alappolinomoknak) nem függnek attól, hogy az a_j súlyok milyen sorrendjéből állítottuk elő a mátrixot továbbá a mátrix utolsó előtti sora

$$P_i[n-1,k] = \begin{cases} \frac{P[n,k]}{1-x}, & \text{ha } k < a_i, \\ \frac{P[n,k]-x\cdot P[n-1,k-a_i]}{1-x}, & \text{ha } k \geq a_i \end{cases}$$

lett volna, ha épp az i játékos lett volna a sorrendben az utolsó. (A képletben feltüntettük az eddig elhanyagolt i-től való függést is). Egy $(1-x)$-el osztható (tetszőleges) polinom

$$\lambda_m \cdot x^m + \lambda_{m-1} \cdot x^{m-1} + \cdots + \lambda_j \cdot x^j + \cdots + \lambda_1 \cdot x + \lambda_0 = \\ = (1-x) \cdot \left(\mu_{m-1} \cdot x^{m-1} + \mu_{m-2} \cdot x^{m-2} + \cdots + \mu_j x^j + \cdots + \mu_0 x\right)$$

alakú felbontásában, egyrész $\sum \lambda_j = 0$, másrész $\mu_j = -(\lambda_m + \lambda_{m-1} + \cdots + \lambda_{j+1})$, így a polinomosztás (és nyilván az összes többi művelet is) elvégezhető $\mathbb{Z}[x]$-en belül (azaz egész számokon értelmezett műveletek körében), $O(n^2)$ időben, azaz a mátrix utolsó sorából az utolsó előtti $O(n^2K)$ lépésben előállítható. A $P_i[n,k]$ polinomok, ugyanazok a polinomok, melyek a 5.3.1. tétel szerint az i játékos Shapley-értékének kiszámítására felhasználhatóak (és a számítás további lépései sem haladják meg az $O(n^2K)$ komplexitást). Mivel ezek a polinomok egymástól függetlenül is előállíthatóak $O(n^2K)$ időben a teljes mátrix nulláról való felépítése nélkül (mely $O(n^3K)$ bonyolultságú, de ezt tehát csak egyszer kell megcsinálni), a teljes számolás különkülön minden $i \in N$ -re elvégezhető egyenként $O(n^2K)$ azaz összesen $O(n^3K)$ idő alatt. Ezzel a 5.3.2. következményt is beláttuk. □

5.3.2. Példa. Illusztrációként tekintsük ismét az 5.3.1. példában vizsgált játékot. Első lépésben előállítjuk a teljes $P[j,k]$ mátrixot, az összes játékosra. Ezt szemlélteti az 5.3. táblázat. A játékosok súlyait önkényesen nagyság szerinti növekvő sorrendben tekintettük, de az 5.3.1. lemma szerint a mátrix utolsó sorában lévő alappolinomok nem függnek ettől a sorrendtől. Ha már az alappolinomok megvannak, a mátrixra többé nincs szükségünk, ezért eleve célszerű úgy előállítani, hogy mindig csak a két aktuális sorát tároljuk. A következő lépést az 5.4. táblázat illusztrálja. Minden egyes játékos Shapley-értéke kiszámolható a táblázat második sorában lévő alappolinomokból. Jelölje a_i az adott játékos súlyát a lineáris reprezentációban. Ekkor $k < a_i$-re a k-adik alappolinomot el kell osztani $(1-x)$-el, jelölje az így kapott polinomot $p_k(x)$ (az i-től

való függést nem jelöljük). Az $(1-x)$-el való osztás szerencsére nagyon egyszerűen elvégezhető: csökkenő x-hatványok szerint rendezve, az új polinom ℓ-edik együtthatója, az eredeti polinom első ℓ együtthatójának az összege, negatív előjellel (feltéve természetesen, hogy a polinom osztható $1-x$-el de ez itt mindig teljesül). Ha $k \geq a_i$ akkor a $p_k(x)$ polinomot úgy kapjuk, hogy a k-adik alappolinomból levonjuk az $x \cdot p_{k-a_i}(x)$ polinomot, és ezt osztjuk $(1-x)$-el. Így kapjuk a táblázat harmadik, hatodik, kilencedik illetve tizenkettedik sorában lévő polinomokat, rendre az első, második, harmadik illetve negyedik játékoshoz (az utolsó játékosnál természetesen visszakaptuk a 5.3. táblázat utolsó előtti, illetve az 5.2. táblázat utolsó sorát). Végül a $p_k(x)$ polinomokat össze kell súlyozni az $f(k+a_i)-f(k)$ súlyokkal, ahol f a játék 5.2.1. definíció szerinti lineáris reprezentációjában lévő függvény (ezeket a súlyokat is feltüntettük a táblázatban közvetlenül a megfelelő $p_k(x)$ polinomok alatt). Az így kapott összesúlyozott polinomok a szaggatott vonalakkal elválasztott szakaszok jobb alsó sarkában vannak feltüntetve. A játék Shapley-értékének kiszámításához ezeket a polinomokat kell kiintegrálni, 0-tól 1-ig, ami jelen példánál

$$
\begin{aligned}
Sh_1 &= \int_0^1 -x^3+4x^2+x\ dx = \left[-\frac{3}{4}x^4+\frac{4}{3}x^3+\frac{1}{2}x^2\right]_0^1 = -\frac{3}{4}+\frac{4}{3}+\frac{1}{2} = \frac{13}{12},\\
Sh_2 &= \int_0^1 -3x^3+4x^2+2x\ dx = \left[-\frac{3}{4}x^4+\frac{4}{3}x^3+x^2\right]_0^1 = -\frac{3}{4}+\frac{4}{3}+1 = \frac{19}{12},\\
Sh_3 &= \int_0^1 -3x^3+4x^2+4x\ dx = \left[-\frac{3}{4}x^4+\frac{4}{3}x^3+2x^2\right]_0^1 = -\frac{3}{4}+\frac{4}{3}+2 = \frac{31}{12},\\
Sh_4 &= \int_0^1 -3x^3+3x^2+7x\ dx = \left[-\frac{3}{4}x^4+x^3+\frac{7}{2}x^2\right]_0^1 = -\frac{3}{4}+1+\frac{7}{2} = \frac{15}{4}.
\end{aligned}
$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$l_1 = 2$	$-x+1$	0	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$l_2 = 3$	x^2-2x+1	0	$-x^2+x$	$-x^2+x$	0	x^2	0	0	0	0	0	0	0	0	0	0	0	0
$l_3 = 5$	$-x^3+3x^2-3x+1$	0	x^3-2x^2+x	x^3-2x^2+x	0	$-x^2+x$	0	$-x^3+x^2$	$-x^3+x^2$	0	x^3	0	0	0	0	0	0	0
$l_4 = 7$	$x^4-4x^3+6x^2-4x+1$	0	$-x^4+3x^3-3x^2+x$	$-x^4+3x^3-3x^2+x$	0	x^3-2x^2+x	0	x^3-2x^2+x	$x^4-2x^3+x^2$	$x^4-2x^3+x^2$	$-x^3+x^2$	0	$-x^3+x^2$	0	$-x^4+x^3$	$-x^4+x^3$	0	x^4

5.3. táblázat. A 5.3.1. példa alappolinomjainak előállítása

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Alappolinomok	$x^4-4x^3+6x^2-4x+1$	0	$-x^4+3x^3-3x^2+x$	$-x^4+3x^3-3x^2+x$	0	x^3-2x^2+x	0	x^3-2x^2+x	$x^4-2x^3+x^2$	$x^4-2x^3+x^2$	$-x^3+x^2$	0	$-x^3+x^2$	0	$-x^4+x^3$	$-x^4+x^3$	0	x^4
$l_1 = 2$	$-x^3+3x^2-3x+1$	0	0	x^3-2x^2+x	0	x^3-2x^2+x	0	x^3-2x^2+x	$-x^3+x^2$	0	$-x^3+x^2$	0	$-x^3+x^2$	0	0	x^3	0	0
$f(k+2)-f(k)$	0	0	0	0	0	0	0	1	2	2	2	2	2	2	2	2	2	2
v_1'																		$-3x^3+4x^2+x$
$l_2 = 3$	$-x^3+3x^2-3x+1$	0	x^3-2x^2+x	0	0	x^3-2x^2+x	0	$-x^2+x$	0	$-x^3+x^2$	0	0	$-x^3+x^2$	0	x^3	0	0	0
$f(k+3)-f(k)$	0	0	0	0	0	0	1	2	3	3	3	3	3	3	3	3	3	3
v_2'																		$-3x^3+4x^2+2x$
$l_3 = 5$	$-x^3+3x^2-3x+1$	0	x^3-2x^2+x	x^3-2x^2+x	0	$-x^3+x^2$	0	x^3-2x^2+x	0	$-x^3+x^2$	$-x^3+x^2$	0	x^3	0	0	0	0	0
$f(k+5)-f(k)$	0	0	0	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5
v_3'																		$-3x^3+4x^2+4x$
$l_3 = 7$	$-x^3+3x^2-3x+1$	0	x^3-2x^2+x	x^3-2x^2+x	0	$-x^2+x$	0	$-x^3+x^2$	$-x^3+x^2$	0	x^3	0	0	0	0	0	0	0
$f(k+7)-f(k)$	0	0	1	2	3	4	5	6	7	7	7	7	7	7	7	7	7	7
v_4'																		$-3x^3+3x^2+7x$

5.4. táblázat. Az alappolinomok visszabontása az algoritmus felgyorsításához

5.3.3. Következmények

Repülőtér-játékok

A fejezet fő eredményeit bebizonyítottuk, azonban arról még nem adtunk számot, hogyan segít az algoritmus az egyébként nagyon fontos játékosztály, a repülőtér-játékok Shapley-értékének kiszámításában. A 3.1. fejezetben bevezetett definíció alapján a repülőtér-játékok karakterisztikus függvénye, $v(S) = \max\{c_i | i \in S\}$, ahol a c_i-k a játékosokhoz rendelt pozitív súlyok, melyeket itt költségként interpretálunk. A maximum függvénytől megszabadulva egy alternatív reprezentációját kaphatjuk a játéknak a következőképpen: tekintsük a játékosoknak a költségek szerinti partícióját és rendezzük ennek osztályait a költségek növekvő sorrendjébe. Tekintsük az osztályokat egy gráf csúcsainak, melyet egészítsünk ki egy s „legkisebb" csúccsal és minden csúcsot kössünk össze a költségek szerint közvetlenül előtte lévővel, a legolcsóbbat az s csúccsal és az élekre írjuk a költségek különbségét (nagyobból a kisebbet levonva).

$$s \xrightarrow{c_1} \underbrace{i_1, i_2, \ldots, i_{\ell_1}}_{c_1=c_2=\cdots=c_{\ell_1}} \xrightarrow{c_{\ell_1+1}-c_{\ell_1}} \underbrace{i_{\ell_1+1}, i_{\ell_1+2}, \ldots, i_{\ell_2}}_{c_{\ell_1+1}=c_{\ell_1+2}=\cdots=c_{\ell_2}} \xrightarrow{c_{\ell_2+1}-c_{\ell_2}} \underbrace{i_{\ell_2+1}, i_{\ell_2+2}, \ldots, i_{\ell_3}}_{c_{\ell_2+1}=c_{\ell_2+2}=\cdots=c_{\ell_3}} \xrightarrow{c_{\ell_3+1}-c_{\ell_3}} \ldots$$

$$\ldots \xrightarrow{c_{\ell_{m-2}+1}-c_{\ell_{m-2}}} \underbrace{i_{\ell_{m-2}+1}, i_{\ell_{m-2}+2}, \ldots, i_{\ell_{m-1}}}_{c_{\ell_{m-2}+1}=c_{\ell_{m-2}+2}=\cdots=c_{\ell_{m-1}}} \xrightarrow{c_{\ell_{m-1}+1}-c_{\ell_{m-1}}} \underbrace{i_{\ell_{m-1}+1}, i_{\ell_{m-1}+2}, \ldots, i_{\ell_m}}_{c_{\ell_{m-1}+1}=c_{\ell_{m-1}+2}=\cdots=c_{\ell_m}}$$

Könnyen látható hogy minden koalíció költsége azon élek költségeinek összege, melyeken a koalíció bármely tagjához el lehet jutni a gráfban az s csúcsból, vagyis a repülőtér-játék előáll egy speciális (egyetlen útból/láncból álló gráffal megadott) költségallokációs játékként, amelyet a 3.2.1. fejezetben tárgyaltunk.

A repülőtér-játékok az 5.2.1. definíció értélmében nem lineáris reprezentációval vannak adva, mert a koalíciók értéke nem a súlyok összegének, hanem azok maximumának függvénye (a költségallokációs reprezentáció sem lineáris reprezentáció, mert a súlyok az élekhez, nem a játékosokhoz vannak rendelve). A matematikai intuíció azt sugallná, hogy ez bonyolítja a dolgokat, mert a nemlineáris függvények általában nehezebben kezelhetőek mint a lineárisak, azonban itt az a helyzet, hogy az 5.3.1. tétel illetve az 5.3.2. tétel bizonyítása során bemutatott séma nem csak hogy erre az esetre is kiválóan használható, hanem az összegzés helyett a maximum függvény használata automatikusan polinomiálissá teszi az értelemszerűen módosított algoritmust.

Legyen i egy tetszőleges játékos és jelölje $c_1, c_2, \ldots, c_m$ az i-től különböző, összes többi repülők költségeit növekvő sorrendben, rendre $\ell_1, \ell_2, \ldots, \ell_m$ multiplicitással, ahol $n = \sum \ell_h$. Jelölje $P[j,k]$ ugyanazt, amit az 5.3.1. tétel bizonyításában, annak a valószínűségét, hogy az i-től különböző játékosok között az első j játékos egy k összköltségű koalíciót alkot, ha egymástól függetlenül x valószínűséggel kooperálnak. Most vegyük észre a következőket:

1. Az 5.3.1. tételnél látotthoz nagyon hasonlóan, a teljes várható érték tételével a $P[j,k]$ polinomok a maximum függvény esetén is felírhatóak rekurzívan: a súlyokat nemcsökkenő sorrendben tekintve, minden sor úgy kapható a felette lévőből, hogy mindenkit megszorzunk $(1-x)$-el, az adott játékos súlyával megegyező indexű oszlopban pedig még hozzáadunk x-et.

2. $P[j,k] = 0$ minden j-re, minden olyan k-ra, amely nem egyezik meg egyik játékos költségével sem. Így a mátrix minden oszlopa, legfeljebb m oszlop kivételével azonosan nulla lesz, melyet sem tárolni sem kiszámolni nem szükséges, a mátrix lényegében $n \times m$-es és polinom-időben feltölthető.

3. A $P[n-1,k]$ polinomok alapján kiszámolható az i játékos Shapley-értéke: a határhozzájárulása egy k költségű koalícióhoz $(c_i - k)^+$, ezekkel a súlyokkal kell a polinomokat szorozni és integrálni, ugyanúgy, mint a lineárisan reprezentálható játékoknál.

Az algoritmus tehát, csekély módosítással a repülőtér-játékokra is alkalmazható és polinom-időben fut, konkrétan $O(n^5)$ időben és $O(n^3)$ tárban, mert a mátrix K helyett csak $m \leq n$ oszlopból áll, azonban minden sor kiszámítása során (a maximum függvény miatt) minden nemnulla elemmel és nem csak kettővel kell számolni.

Mivel a költségelosztási játékok Shapley-értéke lineáris időben számolható, a fenti $O(n^5)$ futási idejű algoritmus látszólag teljesen haszontalan. Vegyük észre azonban, hogy a repülőtér-játékok esetén a $P[j,k]$ mátrixot nem kell rekurzíve kiszámolni, kapásból föl tudjuk írni az utolsó sorát. Legyen c_k egy tetszőleges súly. A $P[j,c_k]$ polinom kezdetben ($j = 0$-ra) 0, majd ezután három dolog történhet vele.

1. Amíg a c_k-nál kisebb költségű játékosokkal számolunk, addig semmi.

2. Amikor a c_k-val megegyező költségű játékosokkal számolunk, akkor megszorozzuk $(1-x)$-el majd hozzáadunk x-et, ilyenkor $P[j+1,c_k] = P[j,c_k] \cdot (1-x) + x$.

3. A c_k-nál nagyobb költségű játékosok esetén megszorozzuk $(1-x)$-el.

Az első és harmadik lépés elég könnyen érthető, a másodikkal kapcsolatban vegyük észre a következőt:

5.3.2. Lemma. *Legyen $p(x) = 0$ polinom és $k \in \mathbb{N}$ egy pozitív egész szám. Ismételjük k-szor a*

$$p(x) := p(x)(1-x) + x$$

műveletet. Ekkor: $p(x) = 1 - (1-x)^k$ lesz.

Bizonyítás. Az állítás $k = 1$-re igaz, mert $1 - (1-x) = x$. k-ról $k+1$-re az

$$\left(1-(1-x)^k\right)(1-x)+x=\left(1-(1-x)^k\right)-x\cdot\left(1-(1-x)^k\right)+x=$$
$$=\left(1-(1-x)^k\right)-x+x\cdot(1-x)^k+x=1-(1-x)^k+x\cdot(1-x)^k=$$
$$=1-\left((1-x)^k-x\cdot(1-x)^k\right)=1-\left((1-x)\cdot(1-x)^k\right)=1-(1-x)^{k+1}$$

átalakítással következik. □

Ha ezt még megszorozzuk annyiszor $(1-x)$-el ahány c_k-nál nagyobb költségű játékos van, akkor a következőt kapjuk:

5.3.3. Következmény.

$$P[n-1,c_k]=\left(1-(1-x)^{\ell_k}\right)(1-x)^{\sum\limits_{c_h>c_k}\ell_h}=(1-x)^{\sum\limits_{c_h>c_k}\ell_h}-(1-x)^{\sum\limits_{c_h\geq c_k}\ell_h}$$

Ez valóban annak a valószínűsége, hogy „a c_k-nál nagyobb költségű játékosok közül senki sem csatlakozik a koalícióhoz, de nem igaz, hogy a legalább c_k költségű játékosok közül senki sem csatlakozik a koalícióhoz (vagyis valaki igen)".

Ezeket a binomiális tétellel kibontva sok tagból álló polinomokat kapunk hatalmas együtthatókkal. Vegyük észre azonban, hogy erre nincs feltétlen szükség. A Shapley-érték kiszámításához ezeket a polinomokat össze kell súlyozni a határ-hozzájárulásokkal,[36] és végül 0-tól 1-ig integrálni. Mivel mind a súlyozás, mint az integrál lineáris, ezeknek a sorrendje felcserélhető, és elvégezhetjük az integrálást ebben a formában is. Egy függvény $[0,1]$-en vett integrálja nem változik, ha függőlegesen tükrözzük az $x=1/2$ egyenesre, azaz az $(1-x)$ helyére x-et írunk. Az integrálás elvégzése után az

$$\int_0^1 P[n-1,c_k]=\int_0^1(1-x)^{\sum\limits_{c_h>c_k}\ell_h}-(1-x)^{\sum\limits_{c_h\geq c_k}\ell_h}=$$
$$=\int_0^1 x^{\sum\limits_{c_h>c_k}\ell_h}-\int_0^1(1-x)^{\sum\limits_{c_h\geq c_k}\ell_h}=\frac{1}{\sum\limits_{c_h>c_k}\ell_h+1}-\frac{1}{\sum\limits_{c_h\geq c_k}\ell_h+1}$$

illetve az

$$\int_0^1 P[n-1,0]=\int_0^1(1-x)^n=\frac{1}{n+1}$$

formulák adódnak.

A kimaradó i játékos Shapley-értékének számolásakor a $c_k<c_i$ tagokat kell súlyozni c_i-c_k hátárhozzájárulásokkal. Ebből egy teleszkópikus összeg adódik, amiből minden tag kiesik, kivéve a legelsőt. Ez azt jelenti, hogy az i játékos minden útszakasz költségének $\frac{1}{\ell+1}$-ed részét állja, ahol ℓ azon játékosok száma akik (rajta kívül) az adott útszakaszt használják, ami konzisztens a költségallokációs játék megoldásával.

[36] Ennél a példánál akár határköltségeknek is nevezhetnénk.

5.3.3. Példa. Tekintsük azt a hat fős repülőtér-játékot, ahol a költségek rendre $A:1$, $B,C:5$, $D:12$, $E:15$ és $F:40$. Ez a

$$s \xrightarrow{1} A \xrightarrow{4} \binom{B}{C} \xrightarrow{7} D \xrightarrow{3} E \xrightarrow{25} F$$

költségelosztási játékkal ekvivalens. Számoljuk ki az F játékos Shapley-értékét. Az adott költségű koalíció létrejöttének valószínűségei rendre

$$P[5,0] = (1-x)^5 \qquad P[5,1] = x(1-x)^4 \qquad P[5,12] = (1-x)^2\left(1-(1-x)^2\right)$$
$$P[5,12] = x(1-x) \qquad P[5,15] = x,$$

az integrálok pedig

$$\int_0^1 P[5,0] = \int_0^1 (1-x)^5 = \int_0^1 x^5 = \frac{1}{6}$$
$$\int_0^1 P[5,1] = \int_0^1 x(1-x)^4 = \int_0^1 (1-(1-x))\,(1-x)^4 = \int_0^1 (1-x)^4 - \int_0^1 (1-x)^5 = \int_0^1 x^4 - \int_0^1 x^5 = \frac{1}{5} - \frac{1}{6}$$
$$\int_0^1 P[5,5] = \int_0^1 (1-x)^2\left(1-(1-x)^2\right) = \int_0^1 x^2(1-x)^2 = \int_0^1 x^2 - \int_0^1 x^4 = \frac{1}{3} - \frac{1}{5}$$
$$\int_0^1 P[5,12] = \int_0^1 x(1-x) = \int_0^1 x - \int_0^1 x^2 = \frac{1}{2} - \frac{1}{3}$$
$$\int_0^1 P[5,15] = \int_0^1 x = \frac{1}{2}.$$

Amiből

$$\begin{aligned} Sh_F = 25 &\cdot \frac{1}{2} \\ &+ (25+3) \cdot \left(\frac{1}{2} - \frac{1}{3}\right) \\ &+ (25+3+7) \cdot \left(\frac{1}{3} - \frac{1}{5}\right) \\ &+ (25+3+7+4) \cdot \left(\frac{1}{5} - \frac{1}{6}\right) \\ &+ (25+3+7+4+1) \cdot \frac{1}{6}. \end{aligned}$$

Az egyenletben a $40-15=25$ határhozzájárulás teljes súlya

$$\frac{1}{2} + \left(\frac{1}{2} - \frac{1}{3}\right) + \left(\frac{1}{3} - \frac{1}{5}\right) + \left(\frac{1}{5} - \frac{1}{6}\right) + \frac{1}{6} = 1,$$

ami helyes, mert a kifutópálya 15 fölötti részét a legnagyobb repülőgép egyedül használja, így neki is kell kifizetni. Az 3 súlya $\left(\frac{1}{2} - \frac{1}{3}\right) + \left(\frac{1}{3} - \frac{1}{5}\right) + \left(\frac{1}{5} - \frac{1}{6}\right) + \frac{1}{6} = \frac{1}{2}$, ami azt jelenti, hogy a 12 és 15 közötti 3 költségű szakasz költségét a két legnagyobb repülő állja, stb. végül az 1 költsége egyenletesen eloszlik a hat játékos között.

Polinomiális esetek

Az 5.3.1. tétel illetve az 5.3.2. tétel szerint lineárisan reprezentálható játékok Shapley-értékének kiszámítása lehet nehéz (akár NP-nehéz is), de ez csupán abból fakad, hogy „túl nagy" számokkal paramétereztük őket. A játékok paramétereire tett egyszerű feltevések mellett az algoritmus polinomiálissá válik. Két ilyen eset például:

5.3.4. Következmény.

1. *Minden lineárisan reprezentálható játékokból álló játékosztályra polinom-időben kiszámolható a Shapley-érték, ha a lineáris reprezentáció súlyai a játékosok számának egy polinomjával korlátozhatók.*

2. *Legyen $\mathcal{G}$ lineárisan reprezentálható játékok olyan osztálya, melyhez létezik egy p polinom, hogy minden $\mathcal{G}$-beli játék esetén $f(k) = c \cdot k \quad ha \quad k \geq p(n)$ alakú ahol f a játék lineáris reprezentációjában szereplő függvény, n pedig a játékosok száma (megjegyzés: a feltevés szerint f függ a játéktól, p viszont nem!). Ekkor $\mathcal{G}$-ben a Shapley-érték polinom-időben kiszámítható.*
   Speciálisan: szavazásos játékok Shapley-értéke polinom-időben kiszámolható, ha a kvóta a játékosok számának egy polinomjával korlátozható.[37]

Bizonyítás. Az első (1) állításon nincs mit bizonyítani, a 5.3.1. tételben a futási időre adott korlát n-ben polinomiális. Ami a (2) állítást illeti, a feltevésből az következik, hogy elegendő a $P[j,k]$ mátrix első $p(n)$ oszlopát kiszámolni, ugyanis az ettől jobbra lévő polinomok már az $f(k+a_i) - f(k) = c \cdot (k+a_i) - c \cdot k = c \cdot a_i$ konstans súlyokkal összegzőknek, így külön-külön nem kell őket ismerni. □

Látens gyorsítás

Repülőtér-játékokra az algoritmus azért polinomiális, mert a $P[j,k]$ mátrixnak legfeljebb polinom-sok oszlopa nem azonosan nulla, és előre meg is tudjuk mondani, hogy melyek ezek. Noha ez a feltevés nem teljesül általában minden lineárisan reprezentálható játékra, célszerű az algoritmust úgy implementálni, hogy ne végezzen felesleges műveleteket exponenciálisan sok nullával. Mivel nem tudjuk előre, hogy melyek lesznek a mátrix nem azonosan nulla oszlopai, az egészet dinamikusan célszerű csinálni, azaz minden lépésben csak az aktuális sor nemnulla elemeivel foglalkozni, és ezek körét folyamatosan bővíteni, ahelyett, hogy előre létrehoznánk egy exponenciálisan sok oszlopból álló nullmátrixot. Ezt úgy lehet legkönnyebben megoldani, ha a $P[j,k]$ mátrix aktuális sorát nem szekvenciálisan egy K hosszú tömbben tároljuk, hanem egy

[37] A tétel kétségtelenül igaz, bár nem túl hasznos, mert a kvóta tipikusan a súlyok összegének felével egyenlő.

dinamikusan bővülő hash-táblában.[38] Ennek persze ára van, mert a hash-függvény (hasítófüggvény, de ez a szóhasználat már magyarul is ritka) kiszámolásának költségei nagyobbak a direkt memóriacímzéshez képest, azonban a mátrix elemeihez való hozzáférés így is (többé-kevésbé[39]) konstans komplexitású, és cserébe csak a tényleg szükséges műveleteket végezzük el. Emellett szükségünk lesz még egy extra lépésre. A 5.3. mátrixot minden nehézség nélkül fölépíthetjük egy hash-táblával, melynek kulcsai a fejlécek, azonban a 5.4. mátrix kiszámolásánál a fejléceken mindenképpen balról jobbra kell haladni, amit csak úgy tudunk megoldani, ha a két lépés között sorbarendezzük a kulcsokat (vagy folyamatosan valamilyen rendezett struktúrában tároljuk, de ennek a költsége ugyanannyi). A rendezés futási ideje $O(K \cdot \log(K))$, ami a súlyokra tett feltevéek szerint $O(Kn)$, ami a teljes algoritmus komplexitását nem változtatja meg, de a másodpercekben mért futási idejét valamelyest rontja. Ha tehát hash-tábla felhasználásával implementáljuk, akkor az algoritmus egy látens faktorral gyorsabb lesz, azaz előre nem tudjuk megmondani, hogy mennyivel, és cserébe a hash-elés és a rendezés miatt kicsit lassabb. Az ellentétes hatások eredője ismeretlen, de sokat biztosan nem rontunk rajta, viszont cserébe a játéktól és a lineáris reprezentáció súlyaitól is függő lehető leggyorsabb algoritmust tudjuk futtatni. Mi több, anélkül, hogy előre tudnánk, az algoritmus komplexitását, az automatikusan olyan gyors lesz, amilyen gyors csak lenni tud, így gyakorlati alkalmazás esetén ez a legjobb, amit tehetünk.

5.3.4. A Shapley-érték becslése Owen módszerével

A 5.1. fejezetben bemutatott multilineáris kiterjesztés (Owen, 1972) segítségével szellemes becslést adhatunk szavazási játékok Shapley-értékére (a részletes leírást ld. pl. Leech, 2003). Legyen i egy tetszőleges játékos, a többi játékos súlyai rendre $w_1, w_2, \ldots w_{n-1}$, és legyen q a kvóta. Az i játékos várható határ-hozzájárulása a 5.3.1. tétel jelöléseivel

$$\sum_{k=q-w_i}^{q-1} P[n-1,k] = \mathbb{P}\left(\mathcal{G}_x \in [q-w_i, q-1]\right)$$

éppen annak a valószínűsége, hogy a többi játékos egy $q-w_i$ és q közé eső szavazóerejű koalíciót alkot. A Shapley-érték ennek az integrálja 0-tól 1-ig. A $\mathcal{G}_x$ valószínűségi változó $n-1$ független tag $\mathcal{G}_x = \sum \lambda_x(w_j)$ alakú összege, melyek x valószínűséggel veszik fel a w_j számokat (és $1-x$ valószínűséggel a nullát), így első két momentuma:

$$\mathbb{E}(\mathcal{G}_x) = \mu_x(w) = x \cdot \sum w_j \qquad \text{illetve} \qquad \mathbb{E}(\mathcal{G}_x^2) = x \cdot \sum w_j^2 + x^2 \sum_{j_1 \neq j_2} w_{j_1} w_{j_2}$$

[38] A hash-table/hash-map vagy magyarul hash-tábla vagy ritka szóhasználattal hasítótábla egy asszociatív tömb, erről bővebben ld. például Cormen et al. (2009) 3. fejezet.

[39] A hash-függvény elméleti legrosszabb esetben akár lineáris is lehet, de átlagban konstans lesz.

szórásnégyzete pedig

$$\sigma_x^2(w) = x\sum w_j^2 - x^2\sum w_j^2 = x(1-x)\sum w_j^2.$$

Ha most a független tagok összegeként előálló $\mathcal{G}_x$ eloszlását normális eloszlással közelítjük, akkor a fenti két momentum meg is határozza a sűrűségfüggvényt. Persze, $\mathcal{G}_x$ nem normális, hanem diszkrét eloszlású, de ha a játékot „elég sok", „elég kicsi" játékos alkotja, akkor a közelítés „elég jó". Így az i játékos várható határ-hozzájárulására a következő becslést nyerjük:

$$\mathbb{P}\left(\mathcal{G}_x \in [q-w_i, q-1]\right) \approx \Phi\left(\frac{q-\mu_x(w)}{\sigma_x(w)}\right) - \Phi\left(\frac{q-w_i-\mu_x(w)}{\sigma_x(w)}\right)$$

ahol Φ a standard normális eloszlás eloszlásfüggvénye. Ezt x szerint numerikusan kiintegrálva megkapjuk az i játékos Shapley-értékének egy becslését.

A módszer értelemszerű módosítással alkalmazható tetszőleges lineárisan reprezentálható játékra. Ehhez a játék lineáris reprezentációjában lévő f függvényt az egész számokról (valamilyen értelmes módon) ki kell terjesztenünk az egész számegyenesre. Az eddig látott példáknál ennek semmi akadálya, de a Shapley-értékre kapott becslésünk függ ettől az önkényesen választott kiterjesztéstől, és elméletileg elképzelhető, hogy a kiterjesztett függvény olyan „idétlenül viselkedik" (két egész szám között össze-vissza ugrál) hogy teljesen elrontja a becslést. Mindenesetre, az i játékos várható határ-hozzájárulását megadó diszkrét formula „folytonosítása" (lineáris helyettesítés után):

$$\sum_{k=0}^{K}\Big(f\left(k+a_i\right)-f\left(k\right)\Big)\cdot\mathbb{P}(\mathcal{G}_x=k) \approx$$

$$\approx \frac{1}{\sigma_x(w)}\int_{-\infty}^{\infty}\left(f\left(\frac{t+a_i-\mu_x(w)}{\sigma_x(w)}\right)-f\left(\frac{t-\mu_x(w)}{\sigma_x(w)}\right)\right)\cdot\varphi(t)\;dt$$

ahol φ a standard normális eloszlás sűrűségfüggvénye. Ezzel a Shapley-értékre egy kettős integrált kapunk (a formulát még ki kell integrálni x szerint is), melynek becslési hibájára nemtriviális felső korlátunk nincs, hiszen a valódi Shapley-érték akár 0 is lehet. Jó hír viszont, hogy a belső integrál felírható analitikusan például a csődjátékok esetén, mivel ezesetben $f(t) = [t-(L-A)]^+$ (ahol L az összes követelés értéke, A pedig az eszközöké) éppen egy call opció[40] kifizetésfüggvénye, amelynek kötési árfolyama a „csőd nagysága", azaz a hiányzó követelések értéke. Az integrálást tagonként elvégezve egy opcióárazási formulára emlékeztető integrált kapunk, ahol az alaptermék árfolyama nem lognormális, hanem normális eloszlású. Az ilyen alakú integrál könnyen kiszámolható, mivel a $t\cdot\varphi(t)$ primitívfüggvénye analítikusan felírható (éppen $-\varphi(t)$), így a Shapley-értékre egy a szavazási játékokéhoz hasonló numerikus integrált kapunk.

[40] Az opció vételi jog, melyről részletes leírás található például Hull (2017) könyvében.

5.4. Korlátok

Az 5.3.1. tétel legfőbb mondanivalója talán (pongyolán fogalmazva), hogy ha egy TU-játék a játékosok számában lineárisan sok paraméterrel adott, akkor a Shapley-érték kiszámításának nehézsége csak abból adódik, hogy a paraméterek túl nagyok. Mivel azt is érszevettük, hogy bármely játék átírható ilyen alakúra (5.2.1. tétel), természetes módon adódik a kérdés, hogy tudjuk-e az eredményeket általánosítani olyan játékosztályokra, melyek eredetileg nem lineáris reprezentációban, hanem valamilyen bonyolultabb alakban vannak megadva. A válasz erre a kérdésre is nemleges. Ando (2012) szerint minimális költségű feszítőfa játékokban (továbbiakban: MCST - minimum cost spanning tree game) a Shapley-érték kiszámítása akkor is #P-nehéz, ha a súlyfüggvény 0 - 1 értékű. Az ilyen játékok a játékosok számában még mindig polinom méretű adathalmazzal vannak megadva, csak a súlyokat nem a játékosokhoz rendeljük, hanem egy gráf éleihez, melynek csúcsai a játékosok. Az eredmény azért is fontos, mert mutatja az 5.3.1. tétel korlátait: hacsak nem P = NP akkor az MCST játékok Shapley-értékére már nincs pszeudopolinomiális algoritmus. Tehát a nem lineáris reprezentációban megadott játékok Shapley-értékét valóban „nehezebb" kiszámolni, mint a lineáris reprezentációban lévőket, mert ezeknél már nem a paraméterek nagysága okozza a nehézséget, itt már találunk ún. erősen NP-nehéz problémákat. Ez nem mond ellent annak, hogy az 5.2.1. tétel szerint az MCST-k is átalakíthatóak lineáris reprezentációjúvá, azonban ennek megvannak a korlátai:

5.4.1. Következmény. *Hacsak nem* P = NP*, akkor az MCST játékok vagy nem lineárisan reprezentálhatóak polinom nagyságú súlyokkal, vagy, ha igen, akkor ez a lineáris reprezentáció nem számolható ki polinom-időben.*

Az állítás nem túl erős, mert nemhogy az nem világos, hogy hogyan lehetne általában egy TU-játék egy nemtriviális lineáris reprezentációját előállítani, de még az sem, hogy ha valaki mond egyet, hogyan tudnánk ellenőrizni, hogy az tényleg az adott játékot reprezentálja. Valószínűleg el kell fogadni, hogy az 5.2.1. tétel ellenére, az 5.3.1. tétel, illetve az 5.3.2. tétel csak olyan játékok Shapley-értékének kiszámításában segít, amelyek „természetüknél fogva" lineárisan reprezentáltak.

Összefoglaló

Ez a dolgozat – sajátos módon – egyszerre szól a Shapley-érték kiszámolásának lehetőségeiről és lehetetlenségeiről (illetve nehézségeiről). Az eredmények matematikai értelemben természetesen konzisztensek, hisz a matematika nem tűri az ellentmondásokat, az intuíció azonban azt sugallja, hogy a probléma egyrészt reménytelenül nehéz, másrészt speciális esetekre jól használható, szép eredmények vannak.

Már a feladat pontos specifikálásánál felmerül két egymást valamilyen értelemben kioltó probléma. Az egyik a játék tárolásának képessége, mert egy általános TU-játék kezelhetetlen méretű adathalmaz, azonban a Shapley-érték ennek az adathalmaznak a függvényében „gyorsan" számolható. Ennek kezelésére bevezettük a polinomiális reprezentáció fogalmát, azaz vizsgálódásainkat olyan játékosztályokra korlátoztuk, melyek valamilyen módon „jól tömöríthetőek", így tudjuk őket tárolni. A másik probláma a Shapley-érték kiszámításának képessége. Ha egy játékosztály elég jól tömöríthető ahhoz, hogy sok játékos esetén is meg tudjuk adni, akkor előfordulhat, hogy ennyi játékosra a Shapley-értéket már nincs kapacitásunk kiszámolni. Ezzel kapcsolatos friss eredményként bizonyítottuk, hogy a tartozásos játékok Shapley-értékének kiszámolása NP-nehéz, és megemlítettünk néhány klasszikus, ismert tételt és következményeit.

A számítástudományi szempontból megoldhatatlannak tűnő problémák kezelésére két lehetőség adja magát. Az egyik a Monte-Carlo szimulációval történő becslés. Ezzel kapcsolatban bemutattunk egy új algoritmust, mely statisztikai módszerekkel képes csökkenteni a becslés hibáját a független mintavételezéshez képest. Ez Shapely-Mann közel hatvan évvel ezelőtt javasolt módszereinek általánosítása, mely speciális játékosztályokra (bár nem mindegyikre) bíztató eredményeket ad. A másik lehetőség az utolsó fejezetben bevezetett lineárisan reprezentálható játékokra kifejlesztett pszeudopolinomiális algoritmus.

A dolgozat, mely – ahogy általában lenni szokott – több kérdést vetett fel, mint amennyit megválaszolt, itt véget ér, de a kutatás nem. Nem nagyon értjük még, hogy a Monte-Carlo módszerek mely játékosztályokon működnek jól, melyeken nem és miért. Egy TU-játék nagyon sokféleképpen megadható a koalíciók értékének felsorolása nélkül is, és sok mindent nem értünk még azzal kapcsolatban, hogyan lehet egyiket a másikból kiszámolni, illetve, hogyan függ et-

től a Shapley-érték komplexitása. Nem tudjuk, hogy van-e polinom-idejű, randomizált, vagy exponenciális, de a brute force módszernél gyorsabb, gyakorlatban használható algoritmus a Shapley-érték kiszámolására vagy becslésére. Ezek mind nehéz kérdések, de talán tettünk egy apró lépést a válaszokhoz vezető úton.

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

Amihud, Y. és Barnea, A. Portfolio selection for managerial control. *Omega*, 2(6):775–783, 1974.

Ando, K. Computation of the Shapley value of minimum cost spanning tree games:# p-hardness and polynomial cases. *Japan Journal of Industrial and Applied Mathematics*, 29(3):385–400, 2012.

Arora, S. és Barak, B. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521424267.

Aziz, H. Computation of the random arrival rule for bankruptcy problems. *Operations Research Letters*, 41(5):499 – 502, 2013. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2013.06.004. URL `http://www.sciencedirect.com/science/article/pii/S0167637713000837`.

Botev, Z. és Ridder, A. Variance reduction. *Wiley StatsRef: Statistics Reference Online*, pages 1–6, 2017.

Castro, J., Gómez, D., és Tejada, J. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. URL `https://doi.org/10.1016/j.cor.2008.04.004`.

Castro, J., Gómez, D., Molina, E., és Tejada, J. Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017. doi: https://doi.org/10.1016/j.cor.2017.01.019. URL `https://doi.org/10.1016/j.cor.2017.01.019`.

Chantreuil, F., Fourrey, K., Lebon, I., és Rebière, T. Decomposing us income inequality à la Shapley: Race matters, but gender too. 2020.

Chun, Y. A new axiomatization of the Shapley value. *Games and Economic Behavior*, 1(2):119–130, 1989. ISSN 0899-8256. doi: https://doi.org/10.1016/0899-8256(89)90014-6. URL `https://www.sciencedirect.com/science/article/pii/0899825689900146`.

Colini-Baldeschi, R., Scarsini, M., és Vaccari, S. Variance allocation and Shapley value. *Methodology and Computing in Applied Probability*, 20(3):919–933, 2018.

Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing - STOC '71*. ACM Press, 1971. doi: 10.1145/800157.805047. URL https://doi.org/10.1145/800157.805047.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., és Stein, C. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

Csóka, P. Az adósságelengedés modellezése kooperatív játékelmélettel. *Közgazdasági Szemle*, 65(7-8):768–779, 2018.

Csóka, P. és Herings, P. J.-J. Liability games. *Games and Economic Behavior*, 116:260–268, 2019. URL https://doi.org/10.1016/j.geb.2019.05.007.

Csóka, P. és Pintér, M. On the impossibility of fair risk allocation. *The BE Journal of Theoretical Economics*, 16(1):143–158, 2016.

Csóka, P., Herings, P. J.-J., és Kóczy, L. Á. Stable allocations of risk. *Games and Economic Behavior*, 67(1):266–276, 2009.

Csóka, P., Illés, F., és Solymosi, T. On the shapley value of liability games. *European Journal of Operational Research*, 300(1):378–386, July 2022. doi: 10.1016/j.ejor.2021.10.012. URL https://doi.org/10.1016/j.ejor.2021.10.012.

de Clippel, G. Membership separability: A new axiomatization of the Shapley value. *Games and Economic Behavior*, 108:125–129, 2018.

Dehez, P. és Ferey, S. How to share joint liability: A cooperative game approach. *Mathematical Social Sciences*, 66(1):44 – 50, 2013. ISSN 0165-4896. doi: https://doi.org/10.1016/j.mathsocsci.2013.02.003.

Deng, X. és Papadimitriou, C. H. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994. URL http://www.jstor.org/stable/3690220.

Driessen, T. S. *Cooperative games, solutions and applications*, volume 3. Springer Science & Business Media, 1988. ISBN 978-90-481-8451-4.

Dubey, P. The Shapley value as aircraft landing fees–revisited. *Management Science*, 28(8):869–874, 1982.

Durstenfeld, R. Algorithm 235: random permutation. *Communications of the ACM*, 7(7):420, 1964.

E. Knuth, D. *The Art of Computer Programming 2: Seminumerical Algorithms*. Addison-Wesley Publ. Company, 1969.

Faigle, U. és Kern, W. On some approximately balanced combinatorial cooperative games. *Zeitschrift für Operations Research*, 38(2):141–152, 1993.

Ferey, S. és Dehez, P. Multiple causation, apportionment, and the Shapley value. *The Journal of Legal Studies*, 45(1):143–171, 2016.

Fisher, R. A., Yates, F., et al. Statistical tables for biological, agricultural aad medical research. *Statistical tables for biological, agricultural aad medical research.*, 1938.

Garey, M. R. és Johnson, D. S. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Harsanyi, J. C. 17. A bargaining model for the cooperative n-person game. In *Contributions to the Theory of Games (AM-40), Volume IV*, pages 325–356. Princeton University Press, December 1959. doi: 10.1515/9781400882168-019. URL `https://doi.org/10.1515/9781400882168-019`.

Harsanyi, J. C. A simplified bargaining model for the n-person cooperative game. *International Economic Review*, 4(2):194–220, 1963.

Hull, J. C. *Options, Futures, and Other Derivatives*. Pearson, Upper Saddle River, NJ, 10 edition, January 2017.

Illés, F. Linearly representable games and pseudo-polynomial calculation of the shapley value. *arXiv preprint arXiv:2205.12324*, 2022.

Illés, F. és Kerényi, P. Estimation of the Shapley value by ergodic sampling. *arXiv preprint arXiv:1906.05224*, 2019.

Karp, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972. doi: 10.1007/978-1-4684-2001-2_9. URL `https://doi.org/10.1007/978-1-4684-2001-2_9`.

Kiss, E. *Bevezetés az algebrába*. Typotex Kft, 2007.

Klinz, B. és Woeginger, G. J. Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences*, 49(1):111 – 116, 2005. ISSN 0165-4896. doi: https://doi.org/10.1016/j.mathsocsci.2004.06.002.

Leech, D. Computing power indices for large voting games. *Management Science*, 49(6):831–837, 2003.

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., és Rogers, A. Bounding the estimation error of sampling-based shapley value approximation, 2013. URL `https://arxiv.org/abs/1306.4265`.

Mann, I. és Shapley, L. S. Values of large games, iv: Evaluating the electoral college by montecarlo techniques. 1960.

Mann, I. és Shapley, L. S. Values of large games, vi: Evaluating the electoral college exactly. 1962.

Markowitz, H. The optimization of a quadratic function subject to linear constraints. Technical report, RAND CORP SANTA MONICA CA, 1955.

Megiddo, N. Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research*, 3(3):189–196, 1978. doi: 10.1287/moor.3.3.189. URL `https://doi.org/10.1287/moor.3.3.189`.

Owen, G. Multilinear extensions of games. *Management Science*, 18(5):P64–P79, 1972. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/2661445`.

Papadimitriou, C. M. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994. ISBN 0201530821.

Peleg, B. és Sudhölter, P. *Introduction to the theory of cooperative games*, volume 34. Springer Science & Business Media, 2007.

Pintér, M. Regressziós játékok. *Szigma*, 38(3-4):131–148, 2007.

Pintér, M. Regression games. *Annals of Operations Research*, 186(1):263–274, 2011.

Pintér, M. Young's axiomatization of the Shapley value: a new proof. *Annals of Operations Research*, 235(1):665–673, 2015.

Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.

Shapley, L. S. és Shubik, M. On market games. *Journal of Economic Theory*, 1(1):9–25, 1969.

Tarashev, N., Tsatsaronis, K., és Borio, C. Risk attribution using the Shapley value: Methodology and policy applications. *Review of Finance*, 20(3):1189–1213, 2016.

Taylor, A. D. és Pacelli, A. M. *Mathematics and politics: strategy, voting, power, and proof.* Springer Science & Business Media, 2008.

von Neumann, J. és Morgenstern, O. *Theory of games and economic behavior.* Princeton University Press, 1944.

Young, H. P. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, 1985.