KOOPERÁCIÓ ÉS GÉPI TANULÁS LABORATÓRIUM

Játékelmélet

Elméleti segédlet

Készítette:
Kovács Dániel László
(dkovacs@mit.bme.hu)

Méréstechnika és Információs Rendszerek Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

2012, október.

Tartalomjegyzék

1. Bevezetés

Jelen segédlet a Budapest Műszaki és Gazdaságtudományi Egyetem Méréstechnika és Információs Rendszerek tanszéke által szervezett „*Intelligens Rendszerek*” M.Sc. szakirány „*Kooperáció és gépi tanulás (VIMIM223)*” c. laboratóriumának „*Játékelmélet*” c. laborgyakorlatához készült. Célja a hallgatók gyakorlatra való elméleti és gyakorlati felkészítése.

Az anyagban először a Neumann János féle Játékelmélet [1] alapvetéseivel ismerkedhetünk meg, majd pedig a kapcsolódó laborgyakorlat szoftver infrastruktúrájával. Az anyagot irodalomjegyzék, illetve példa-játékok gyűjteménye zárja.

2. Játékelméleti alapvetések

A laborgyakorlatra való felkészülés során célszerű megértenünk a főbb játékelméleti [1] alapfogalmakat, hiszen a gyakorlat során „élesben” fogjuk használni őket.

Mit is jelent például a „játék”? – Játékosok összessége? Szabályok összessége?

Esetünkben szerencsére ezek a kifejezések egyáltalán nem ködösek.

Definíció (Játék): Játéknak nevezzük a játékosok (ágensek) halmazát, a játékosok lehetséges stratégia-halmazainak halmazát, és az egyes játékosokhoz tartozó stratégia-halmazokból képezett Descartes-i szorzatokhoz valós számértéket rendelő haszonfüggvények halmazát egybefogó 3-ast.

Tehát általánosságban van *n* darab ágensünk, akiket most játékosnak nevezünk. Mindegyik játékosnak van 1-1 stratégia-halmaza, és 1-1 haszonfüggvénye. A játékosok stratégia-halmaza tartalmazza azokat a tiszta stratégiákat, amik szerint az adott játékosok a játék egy-egy lejátszása (játszmája) során játszhatnak.[1] Amennyiben egy adott lejátszás alkalmával minden játékos választott magának 1-1 stratégiát a saját stratégia-halmazából, egy stratégia-kombináció alakul ki (a stratégia halmazok Descartes-i szorzatának egy eleme). Ezekhez társít egy-egy valós számértéket, egy-egy hasznot a játékosok haszonfüggvénye. Minden játékosnak van tehát 1-1 saját haszonfüggvénye, amivel értékeli a stratégia-kombinációkat, pontosabban a stratégia-kombinációk eredményeképp előálló kimeneteleket.

Formálisan az előbbieket a következőképp foglalhatjuk össze: a játék egy 3-as: $\Gamma = (N, \{S_i\}_{i\in N}, \{u_i\}_{i\in N})$, ahol $N = \{1,2,\dots,n\}$ a játékosok halmaza, S_i az *i* játékos tiszta stratégiáinak halmaza, $u_i : S \to R$ pedig az *i* játékos haszonfüggvénye, amely a stratégia-kombinációk $S = S_1 \times S_2 \times \cdots \times S_n = \times_{i\in N} S_i$ halmazából (azaz a lehetséges kimenetelekből) képez a valós számok *R* halmazába (azaz meghatározza, hogy mennyire jó egy-egy kimenetel az *i* játékos számára).

Játék

$$\Gamma = \left(N, \{S_i\}_{i\in N}, \{u_i\}_{i\in N}\right) \qquad N = \{1, 2, \dots, n\}$$

Stratégiák és kombinációik

$$S = S_1 \times S_2 \times \cdots \times S_n = \times \prod_{i=1}^{n} S_i$$

$$S = \{(s_1, s_2, \dots, s_n) \mid s_1 \in S_1, s_2 \in S_2, \dots, s_n \in S_n\}$$

Haszonfüggvények

$$\forall i \in N - re \quad u_i : S \to \Re$$

Természetesen ideális/racionális esetben minden játékos arra törekszik, hogy maximalizálja a haszonfüggvényét (azaz saját hasznát). Viszont ennek módja egyáltalán nem nyilvánvaló, és épp ez a játékelmélet egyik kulcsmomentuma. Az egyes játékosok ugyanis csak a saját stratégiájukat tudják megválasztani. Mindenki külön-külön választ stratégiát. Ráadásul egymás választásának ismerete

[1] Fontos különbséget tennünk a „játék” és a „játszma” között. Egy-egy játéknak nyilván többféle lejátszása lehet, azaz többféle játszma is adódhat egy játék szabályai szerint. Különböző játszmák, a játék mégis ugyanaz…

nélkül, mondhatni egyszerre kell meghozniuk a döntést. Az egyes játékosok tehát a játszma kezdetén nem tudják, hogy a többiek milyen stratégia szerint fognak játszani – ez úgyis csak a lejátszás végén derül csak ki.

Vegyük például a jól ismert Fogolydilemma nevű játékot (lásd. 5/B szakasz). A játékban 2 játékos szerepel, mindkettőnek 2-2 stratégiája van: „Vall", illetve „Hallgat". Egymástól függetlenül, mindenféle kommunikáció nélkül kénytelenek stratégiát választani. A játék kimenetele (az egyes játékosok haszna/kifizetése) pedig akkor adódik, mikor már mindketten választottak.

Ha esetleg kicsit erőltetettnek tűnik ez a modell (pl. hogy egyszerre kell választani, a többiek választását nem ismerve), akkor tartsuk szem előtt, hogy egyelőre elvben még vajmi kevés ismerettel rendelkezünk a Játékelméletről, és nem volna célszerű elhamarkodottan képet alkotnunk róla. A Játékelmélet ugyanis, még ilyen alapvető formájában is, számos valós világbeli helyzet leírására/modellezésére alkalmas.

A modell, amiről eddig beszéltünk, az úgynevezett „normál alak". A játékoknak van azonban egy úgynevezett „extenzív alakjuk" is, amely már jóval mélyebb/részletesebb bepillantást enged a stratégiák szerkezetébe, a környezet/játék szabályszerűségeibe, a játékosok által birtokolt ismeretekbe, a játékmenet (azaz játszma) dinamikájába, temporális és logikai felépítésébe, stb. …és mindez még csak a klasszikus játékelmélet alapvető eszköztára. A modern játékelmélet, mint például a nem-teljes információjú játékok [2], az evolúciós játékelmélet [3], a mechanizmus-tervezés (inverz játékelmélet) [4] és társai, rengeteg további hasznos újdonságot hoztak be.

Most azonban egyelőre elégedjünk meg a játékok normál alakjának bemutatásával. Példának okáért itt van a Fogolydilemma. *Miért dilemma ez?* – Próbáljunk meg belegondolni, hogy mit tennénk, ha valamely játékos helyében volnánk! Bizonyára érezhető, hogy a döntés (hogy milyen stratégia szerint cselekedjünk) egyáltalán nem triviális, még egy ilyen „egyszerű" játék esetében sem.

A Fogolydilemma esetén látható, hogy ha mindketten vallunk, akkor rosszabbul járunk, mint ha mindketten hallgatnánk, nyilván. *Mégis, megkockáztassuk akkor a hallgatást?* Akár 10 évünkbe is kerülhet (ha a másik játékos viszont vall). A Fogolydilemmában ezért az ésszerű döntés az, ha a vallunk. Nem ismerjük a másik játékost, nem tudjuk, hogy mit fog tenni, nem tudunk kommunikálni/koordinálni/kooperálni, és előzetes megállapodás és/vagy bizalom sincs közöttünk. Ezért nevezik az előbbi játékokat vizsgáló elmélet általában nem-kooperatív játékelméletnek.

De miért is ésszerű vallani a Fogolydilemmában? – Vegyük észre, hogy ha vallunk, úgy minden esetben jobban járunk, mint hogyha bármi mást tennénk (azaz hallgatnánk). Ráadásul ez mindkét játékosra igaz. Ezt nevezik dominanciának – a Fogolydilemmában a „Vall" stratégia dominálja a „Hallgat" stratégiát mindkét játékos esetén, és így mindkettőjük esetén ez a domináns stratégia.

Definíció (Domináns stratégia): egy játékos valamely stratégiáját dominánsnak nevezzük, ha a többi játékos döntésétől függetlenül, minden esetben jobb eredményt (magasabb kifizetést) ad, mint a játékos többi stratégiája.

Dominancia (gyenge)

Az $i \in N$ játékos $s_i, s_i^* \in S_i$ stratégiái esetén $s_i^* \succ s_i$, ha

$\forall s_{-i} = (s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n) \in S_{-i} = \underset{\substack{j \in N \\ j \neq i}}{\times} S_j$ esetén

$u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ teljesül, továbbá

$\exists s'_{-i} \in S_{-i}$ úgy, hogy $u_i(s_i^*, s'_{-i}) > u_i(s_i, s'_{-i})$

Tehát a Fogolydilemmában mindkét játékos domináns stratégiája a „Vall”, a játék ugyanis szimmetrikus, azaz mindkét játékosnak ugyanaz a stratégia-halmaza, és ha bárhogyan helyet cserélnének (másik játékos szerepébe bújnának), akkor sem változna saját szemszögükből nézve az egyes kimenetelek egyéni haszna.

Szimmetria

A játék szimmetrikus, ha $S_1 = S_2 = \cdots = S_n$, és

$\forall i \in N$, $s \in S$, és π permutáció esetén

$$u_i(s_1, s_2, \ldots, s_n) = u_{\pi(i)}\left(s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)}\right)$$

Ha $n = 2$, akkor $S_1 = S_2$, és $\forall s \in S$-re $u_1(s_1, s_2) = u_2(s_2, s_1)$

Mi tehát egy játék megoldása? - Tekintsük a többi játékos összes lehetséges stratégia-kombinációját, és nézzük meg, hogy van-e egy olyan stratégiánk, amely minden esetben jobb, mint a többi stratégiánk? Ha van ilyen stratégiánk, akkor bízvást racionálisnak nevezhetjük, mert hiszen mindig jobb eredményt ad, mint a többi. Ekkor ésszerű ezt választanunk. Sajnos azonban nem minden játékban van domináns stratégiája a játékosoknak (lásd. pl. a 5/C szakaszban bemutatott „*Gyáva Nyúl (Chicken)*” nevezetű játékot).

A domináns stratégiák választásának elve igazából nem veszi figyelembe a többi játékos hasznát (sem önző módon, sem pedig önzetlenül), márpedig végső soron a többi játékos döntésétől is függ, hogy mennyi lesz a mi hasznunk a végén! Foglalkoznunk kell tehát az ő motivációikkal is (a sajátjainkon túl). A többi játékos motivációját pedig lényegében a különböző kimenetelekhez rendelt haszonértékük határozza meg. Erre ad megoldást a Nash-egyensúly [5]:

Definíció (Nash-egyensúly): Nash-egyensúlynak nevezzük azt a stratégia-kombinációt, amelytől egyik játékosnak sem érné meg egyoldalúan eltérnie, ha aszerint történne a lejátszás.

Tehát itt arról van szó, hogy alapesetben olyan kimenetelt keresünk a játékban (2-szereplős játék esetén cellát a játék mátrixában), amelyre teljesül, hogy olyan kifizetések szerepelnek benne az egyes játékosok számára, amiknél nem kaphatnának nagyobbat, ha egyoldalúan stratégiát változtatnának. Tehát azért játszunk Nash-egyensúly szerint, mert ha így játszunk, akkor nincs értelme másképp játszani.[2]

Nash-egyensúly

$s^* = (s_1^*, s_2^*, \ldots, s_n^*) \in S$ stratégia-kombináció NE, ha

$\forall i \in N$ és $s_i \in S_i$ esetén $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$

Természetesen, ha többen összefognak, és együttesen változtatnak stratégiát, az már más kérdés. *Nézzük csak meg újra a Fogolydilemmát! Hol van ott Nash-egyensúly? A táblázat mely mezőjére teljesül, hogy onnan már senkinek sem érné meg eltérnie (ha az a kimenetel adódna egy lejátszásban)?*

[2] A Nash-egyensúly ilyen értelemben tehát, főleg ha unikális, olyan, mint egy „önbeteljesítő jóslat”…

Rövid gondolkodás után beláthatjuk, hogy a „Vall-Vall” stratégia-kombináció által azonosított esetben teljesül a Nash-egyensúly feltétele. Ha ebben az esetben bármely játékos eltérne, azaz stratégiát változtatna, nem járna jobban (sőt, rosszabbul járna).

Ha viszont együttesen változtathatnának stratégiát (esetleg valamiféle közösen osztott irányelv (pl. betyárbecsület, altruizmus, önzetlenség, Kant-i kategorikus imperatívusz) miatt, vagy előzetes egyeztetés nyomán, kölcsönös bizalomra építve), akkor akár még az összességében legjobb, „Hallgat-Hallgat” kimenetelnél is kiköthetnének.

Miért nem kötnek ki ott? Miért vallunk, ha hallgathatnánk is? Tényleg annyira kockázatos?

A Fogolydilemma, mint látjuk, nem kooperatív játék. Kizárja a kooperációt. Alapfeltevése, hogy a játékosok nem kooperálnak, és mindenki egyéni hasznának maximalizálására törekszik. Ebből kifolyólag tehát nem meglepő, ha mindkét játékos a domináns stratégiáját választja, és vall…

Viszont például az 5/F szakaszban bemutatott játéknak mi a Nash-egyensúlya?

A válasz: hiába keressük, mert tiszta stratégiák esetén az „Érmepárosítás (Matching Pennies)” nevezetű játéknak nincs Nash-egyensúlya. *Mégis, mi lehet ekkor a játék megoldása?*

A probléma megoldását a kevert stratégiák (mixed strategies) bevezetése adja:

Definíció (Kevert stratégia): Kevert stratégiának nevezzük a játékosok tiszta stratégiái felett értelmezett valószínűség-eloszlásokat, amik meghatározzák, hogy melyik tiszta stratégiát milyen valószínűséggel játsszák.

Formálisan ez a következőképpen néz ki: minden *i* játékos esetén bevezethetünk egy $Q_i = \Delta(S_i)$ halmazt, amely az *i* játékos S_i tiszta stratégiái felett értelmezett összes lehetséges valószínűség-eloszlás halmaza. Ennek egy-egy $q_i \in Q_i$ eleme az *i* játékos egy-egy kevert stratégiája, azaz egy valószínűség-eloszlás, amely az *i* játékos minden $s_i \in S_i$ tiszta stratégiájához rendel egy-egy $q_i(s_i)$ valószínűséget ([0,1]-beli valós számértéket), ahol tehát $0 \leq q_i(s_i) \leq 1$, és $\sum_{s_i \in S_i} q_i(s_i) = 1$. Ennek megfelelően a játékosok haszonfüggvényének definícióját is célszerű revideálni egy picit: kevert stratégiák esetén $u_i: Q \to R$ az *i* játékos haszonfüggvénye, ahol a $Q = Q_1 \times Q_2 \times \cdots \times Q_n = \times_{i \in N} Q_i$ a kevert stratégia-kombinációk halmazát (azaz a lehetséges kevert-kimeneteleket) jelöli. Egy kevert-stratégia kombináció esetén tehát a játékosoknak már csak várható hasznáról beszélhetünk.

Kevert stratégiák

$i \in N$ egy kevert stratégiája $q_i \in Q_i = \Delta(S_i)$, ahol

$\forall s_i \in S_i$-re $q_i(s_i) \geq 0$, és $\sum_{s_i \in S_i} q_i(s_i) = 1$ (tiszta: $q_i(s_i) = 1$)

Kevert stratégia-kombinációk

$q = (q_1, q_2, \ldots, q_n) \in Q = Q_1 \times Q_2 \times \cdots \times Q_n$

Haszonfüggvények

$\forall i \in N$ – re $u_i : Q \to \Re$, ahol $q \in Q$ esetén

$$u_i(q) = \sum_{s=(s_1, s_2, \ldots, s_n) \in S} q_1(s_1) \cdot q_2(s_2) \cdot \ldots \cdot q_n(s_n) \cdot u_i(s)$$

John F. Nash híres (egzisztenciális, 1951-ből származó) tétele [5] azt mondja ki, hogy kevert stratégiák esetén mindig létezik Nash-egyensúly. Magyarán még az „Érmepárosítás (Matching Pennies)" játéknak is van Nash-egyensúlya, ha megengedjük azt, hogy a játékosok kevert-stratégiák szerint játsszanak/válasszanak tiszta stratégiát.

Kevert Nash-egyensúly

$q^* = (q_1^*, q_2^*, \ldots, q_n^*) \in Q$ kevert NE, ha

$\forall i \in N$ és $q_i \in Q_i$ esetén $u_i(q_i^*, q_{-i}^*) \geq u_i(q_i, q_{-i}^*)$, ahol

$q_{-i}^* = (q_1^*, q_2^*, \ldots, q_{i-1}^*, q_{i+1}^*, \ldots, q_n^*)$ Legjobb válasz?

Azaz $\forall i$-re $q_i^* \in \arg\max_{q_i \in Q_i} u_i(q_1^*, q_2^*, \ldots, q_{i-1}^*, q_i, q_{i+1}^*, \ldots, q_n^*)$

Vajon mi az „Érmepárosítás (Matching Pennies)" játék Nash-egyensúlya?

…természetesen nem csak a domináns és/vagy Nash-egyensúlyhoz hasonló nem-kooperatív elven választhatnak a játékosok stratégiát. A gyakorlatilag végtelen sok lehetőség (random, konstans, stb) közül például törekedhetnek kooperativitásra is. Erre mutat példát a Pareto-optimum:

Definíció (Pareto optimum): Pareto-optimumnak nevezzük azt a stratégia-kombinációt, amelytől senkinek sem érné meg egyoldalúan eltérnie úgy, hogy közben emiatt senki más nem jár rosszabbul.

A Pareto-optimum tehát, mint elv, az előbbiekkel ellenben már bizonyos fokig kooperatív. Látható, hogy a többiek helyzetével kapcsolatos „empátia" is megjelenik benne. A Fogolydilemma esetén például több ilyen stratégia-kombináció is van – a „Vall-Vall" kivételével mindegyik. De viszont ezek közül továbbra is mindegyik „érzékeny" az önzésre. A Fogolydilemma esetén tehát mindegyik Pareto-optimális esetben, mint látjuk, van olyan játékos, akinek megérné egyoldalúan eltérnie attól a kimeneteltől. Épp ezért nem nem-kooperatív/önző/önérdekű játékosok nem fogják ezeket a kimeneteleket választani, sajnos.

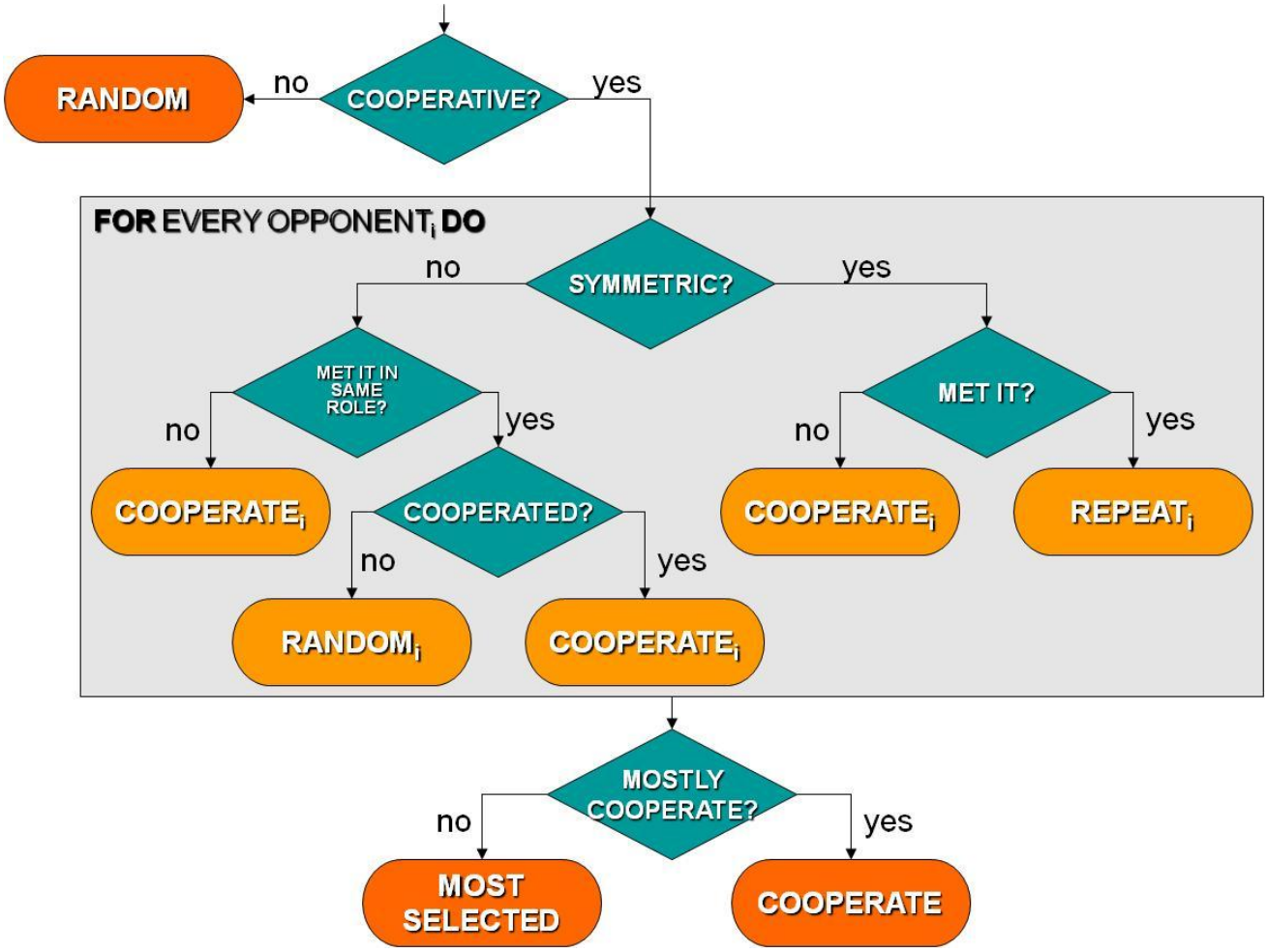
Eddig mindvégig csak „egylövetű" játékokkal foglalkoztunk. *Mi lenne, ha egy játékot adott esetben egymás után többször is játszhatnánk valakivel? Mi lenne, ha nem csak egy játszma lenne egymás után, hanem több? Befolyásolná ez a döntésünket, vagy minden játszmában, vakon/mindig ugyanúgy cselekednénk?*

A válasz (általánosságban): nyilván nem. Az ismételt játékok (repeated games) tanulmányozása is fontos része a klasszikus játékelméletnek. Számos esetben előfordul, hogy egy játékot ismételt módon, egymás után többször is lejátszunk valakivel, és bizony akkor érdemes figyelembe venni azt, hogy mi történt előzőleg (ki-mikor-mit tett). Egy jó stratégia lehet például a következő (2-játékos esetén):

Definíció (Szemet-szemért – Tit for Tat – stratégia): ismételt 2-szereplős játékok esetén azt nevezzük szemet-szemért stratégiának, mikor egy játékos kezdetben kooperál, majd a következő körökben mindig azt cselekszi, amit ellenfele az előző körben tett.

A „Tit for Tat" stratégia [6], a maga egyszerűsége ellenére, hatalmas adaptivitást biztosíthat az előbbiekben taglalt „statikus" irányelvekhez képest (az ismételt játékokban). Érdemes megfontolni!

De miért is beszélünk itt most ismételt játékokról? – Nyilván azért, mert a laborgyakorlaton is fogunk velük foglalkozni. Ráadásul a laborgyakorlat során N-szereplős játékokkal foglalkozunk, amikre az alapjában 2-szereplős játékokhoz kigondolt TFT stratégia nem alkalmazható, viszont megfelelőképpen általánosítva ez sem jelent problémát [7].



1. ábra: N-szereplős játékokhoz általánosított TFT stratégia [7]

3. A laborgyakorlat ágensei

A jelen segédlethez kapcsolódó laborgyakorlat során JADEX-es [8], és szokványos JADE-es [9] ágensekkel dolgozunk. Három különböző fajtájú ágenssel lesz dolgunk:

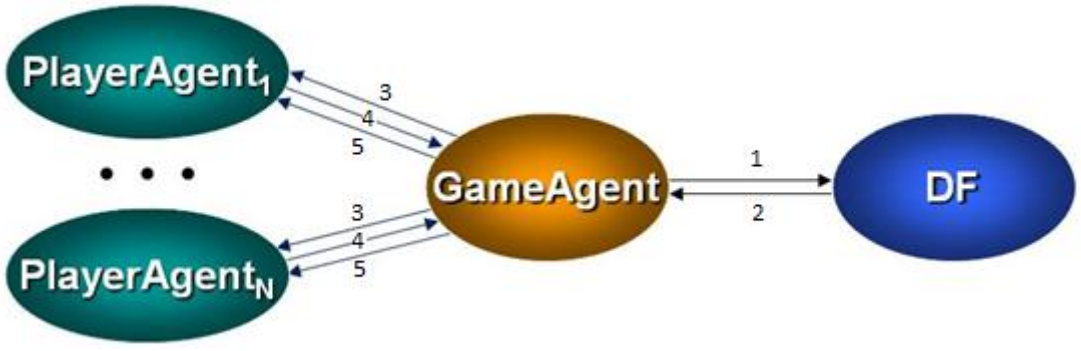
1. **Játékvezető-ágens:** `msclab01.gametheory_lab.GameAgent.GameAgent`
2. **Játékos-ágens:** `msclab01.gametheory_lab.PlayerAgent.Player`
3. **Felhasználói-ágens:** `msclab01.gametheory_lab.UserAgent.UserAgent`

Az említett ágensek működése röviden a következő: a `GameAgent` ágens egy adott típusú, körökre osztott, ismételt játékot host-ol. Indítás után elkezdi keresni a platformon található játékos és/vagy felhasználói ágenseket (elkéri a DF ágenstől a listájukat). A játékosok és a felhasználók közötti különbség, hogy az előbbi gépi (`PlayerAgent`), míg az utóbbi emberi (`UserAgent`) játékost takar. Az utóbbi fajtájú wrapper-ágens arra szolgál, hogy emberi felhasználók is becsatlakozhassanak a platformon zajló játszmákba.

Miután a `GameAgent` ágens legalább N db. megfelelő ágenst talált (N-szereplős játék esetén), küld feléjük egy REQUEST üzenetet, melyben közli, hogy ki, kivel, és milyen szerepben játszik az adott körben. A játékosok, miután megkapják ezt az értesítést, egy-egy INFORM üzenettel válaszolnak, amiben közlik többek között a választott stratégiájukat.

A `GameAgent` ágens, miután megkapja az adott körben játszó összes játékostól a választ, összesíti az eredményeket, majd pedig személyre szabottan kihirdeti az eredményt (a kifizetéseket) minden egyes játékos/felhasználói ágens számára egy INFORM üzenet formájában A játékosok ennek hatására felülvizsgálják saját belső állapotuk (Belief-Revision, GUI update, stb), majd amennyiben hasznuk a minimum alá csökkent, elpusztulnak (deregisztrálnak a DF ágensnél, és kilépnek a platformról), egyébként pedig újabb játszmára felszólító REQUEST üzenetre várnak.

A `GameAgent` ágens tehát a fentieket ismétli újra és újra annyiszor, ahány kör van a játékban, vagy amíg az ágensek száma meg nem haladja a maximumot (ha pl. a játékos-ágensek számára engedélyezzük azt, hogy adott kumulált haszon fölött osztódjanak[3]), vagy N alá nem csökken.



2. ábra: a laborgyakorlat ágenseinek együttműködése 1-1 játékkörben (vázlatosan)

A `Player` ágenseknek, lévén autonóm módon hozzák meg döntéseiket, különböző típusai lehetnek attól függően, hogy milyen módon választanak játékstratégiát. Az alapértelmezett típusokon felül (adott/konstans stratégia szerint játszó, véletlenszerű stratégia szerint játszó, kooperatív, nem-

[3] A jelen segédlethez kacsolódó laborgyakorlat során nem foglalkozunk evolúcióval, így osztódásról sem lesz szó.

kooperatív, domináns stratégia szerint játszó, Nash-egyensúly szerint játszó, Pareto-optimum szerint játszó, stb) számos saját típust definiálhatunk.
Az ágenseknek (mindhárom fajtának) többféle paramétere van/lehet. A `GameAgent`, és a `UserAgent` ágens szokványos JADE-es ágens, tehát indításuk is ennek megfelelő. A következő – pirossal jelölt nevű – paramétereik vannak:

- `GameAgent`
  - **Játék azonosítója**
    - Típusa: `String`
    - Opcionális (de kötelező, ha a második argumentumot is meg szeretnénk adni)
    - Alapértelmezett értéke (ha nincs megadva): `Game.HAWK_DOVE` konstans
    - Értékkészlete: az `msclab01.gametheory_lab.Game` osztály konstansai, vagy a `\jade\src\msclab01\gametheory_lab\games` könyvtárban található `.nfg` kiterjesztésű GAMBIT-es normál formájú játékok fájlneve (kiterjesztés nélkül)
    - Leírás: *arra szolgál, hogy a `GameAgent` ágens induláskor automatikusan be tudja azonosítani, hogy milyen típusú játékot kell host-olnia.*
  - **Játékosok maximális száma (körönként)**
    - Típusa: `int`
    - Opcionális (de csak az első argumentummal együtt lehet megadni)
    - Alapértelmezett értéke (ha nincs megadva): `500`
    - Értékkészlete: tetszőleges pozitív egész szám (legalább 2)
    - Leírás: *arra szolgál, hogy a `GameAgent` ágens tudja, hogy a `DF` ágens által egyes körökben visszaadott, potenciális játékosokat tartalmazó találatok közül legfeljebb mennyit választhat ki (véletlenszerűen, egyenletes eloszlás szerint) párba-rendezés és játék céljából. Tehát legfeljebb annyi játékos lesz körönként, amennyit itt megadunk.*
  - **Grafikus felhasználói felület (GUI) típusa**
    - Típusa: `int`
    - Opcionális (de csak az első két argumentummal együtt lehet megadni)
    - Alapértelmezett értéke (ha nincs megadva): `GameAgentGui.GUI_PLAYERS_UTIL` konstans
    - Értékkészlete: az `msclab01.gametheory_lab.GameAgent.GameAgentGui` osztály `GUI_...` kezdetű konstansai
    - Leírás: *azt határozza meg, hogy a `GameAgent` ágens miképpen, milyen fajtájú grafikonon jeleníti meg a játékmenetet.*

Ezen felül még 2 lényegesebb paraméter van az ágens kódjának elején:

```
/** A körök maximális száma */
private int max_rounds        = 250;
/** A játék megkezdése előtt ennyi miliszekundumon át várunk */
private int wait_to_start     = 5000;
```

Példa egy `GameAgent` ágens indítására (Héja-Galamb játék, max. 800 ágens, ágensek száma grafikon):

```
java jade.Boot -container ga:msclab01.gametheory_lab.GameAgent.GameAgent(0 800 3)
```

- `UserAgent`
  - **Játék azonosítója**
    - Típusa: `String`
    - Opcionális (de kötelező, ha a második argumentumot is meg szeretnénk adni)
    - Alapértelmezett értéke (ha nincs megadva): `Game.HAWK_DOVE` konstans
    - Értékkészlete: az `msclab01.gametheory_lab.Game` osztály konstansai, vagy a `\jade\src\msclab01\gametheory_lab\games` könyvtárban található `.nfg` kiterjesztésű GAMBIT-es [10] normál formájú játékok kiterjesztés nélküli neve
    - Leírás: *arra szolgál, hogy a* `UserAgent` *ágens induláskor automatikusan be tudja azonosítani, hogy milyen típusú játékban fog körről-körre részt venni.*
  - **Kezdeti haszon**
    - Típusa: `double`
    - Opcionális (de csak az első argumentummal együtt lehet megadni)
    - Alapértelmezett értéke (ha nincs megadva): `0.0`
    - Értékkészlete: zérusnál nem kisebb valós számok
    - Leírás: *az ágens kiindulási összhaszna.*

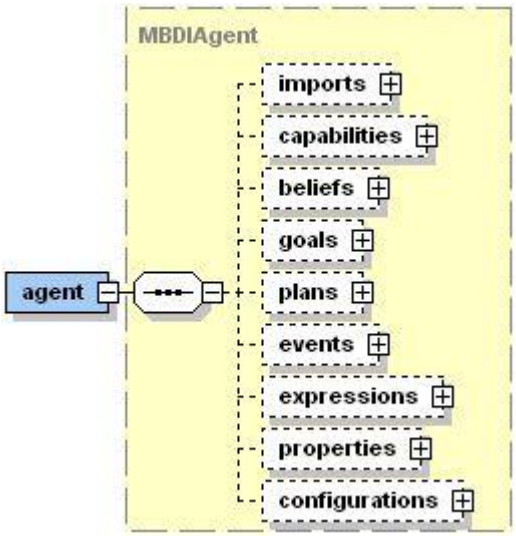
Példa egy `UserAgent` ágens indítására (Héja-Galamb játék, 1.2 kiindulási összhaszon):

```
java jade.Boot -container ua:msclab01.gametheory_lab.UserAgent.UserAgent(0 1.2)
```

Amint fentebb is írtuk, a gépi játékost megvalósító `Player` ágens már nem egy szokványos JADE-es ágens, hanem egy JADEX-es ágens, melynek a

```
\jade\src\msclab01\gametheory_lab\PlayerAgent\Player.agent.xml
```

fájl tartalmazza a leírását. Ennek a leírásnak lényegében 9 főbb (jórészt opcionális) eleme van. Ezeket sorolja fel a következő ábra.



3. ábra: a JADEX ágenseket leíró ADF fájlok vázlatos XML sémája (XSD-je)

Az ADF-ek felépítéséről, és a JADEX-ről bővebben a következő elérésen olvashatunk:

http://vsis-www.informatik.uni-hamburg.de/projects/jadex/jadex-0.96x/doc_overview.php

Most térjünk vissza a JADEX-es `Player` ágens paraméterezésének, és indításának ismertetésére. Legfeljebb 8 darab különböző paramétert adhatunk meg az ágensnek. Ezek a következők.

- `Player`
  - **`gui`**
    - Típusa: `boolean`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `false`
    - Értékkészlete: `true` vagy `false`
    - Leírás: *ettől függ, hogy megjelenítődik-e a* `Player` *ágens GUI-ja, vagy sem. Utóbbi esetben nyilván gyorsabb a működés (főleg nagyon sok ágens esetén).*
  - **`gid`**
    - Típusa: `String`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `Game.HAWK_DOVE` konstans
    - Értékkészlete: az `msclab01.gametheory_lab.Game` osztály konstansai, vagy a `\jade\src\msclab01\gametheory_lab\games` könyvtárban található `.nfg` kiterjesztésű GAMBIT-es normál formájú játékok fájlneve (kiterjesztés nélkül), ez utóbbi String-ként megadva, azaz escape-karakterekkel, pl. `gid=\\\"pd\\\"`
    - Leírás: *arra szolgál, hogy a* `Player` *ágens induláskor automatikusan be tudja azonosítani, hogy milyen típusú játékban fog részt venni körről-körre. Mondhatni, ez a játékról alkotott modellje/hiedelme.*
  - **`myType`**
    - Típusa: `int`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `PlayerType.RANDOM` konstans
    - Értékkészlete: az `msclab01.gametheory_lab.PlayerType` osztály konstansai
    - Leírás: *arra szolgál, hogy megadjuk, hogy játék közben milyen elv/program szerint válasszon stratégiát a* `Player` *ágens.*
  - **`utility`**
    - Típusa: `double`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `0.0`
    - Értékkészlete: tetszőleges zérusnál nem kisebb valós szám
    - Leírás: *arra szolgál, hogy megadjuk, hogy a játékba történő belépéskor, az első kör kezdetén milyen haszonnal rendelkezzen a* `Player` *ágens.*
  - **`max_reproduction_num`**
    - Típusa: `int`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `0`
    - Értékkészlete: tetszőleges egész szám, de legalább `0`
    - Leírás: *arra szolgál, hogy megadjuk, hogy a játék/futtatás során összesen hányszor reprodukálódhat a* `Player` *ágens.*
  - **`reproduction_cost`**
    - Típusa: `double`
    - Teljesen opcionális
    - Alapértelmezett értéke (ha nincs megadva): `20.0`
    - Értékkészlete: tetszőleges pozitív egész szám (nem lehet zérus)
    - Leírás: *arra szolgál, hogy megadjuk, hogy a játék során legalább mennyi hasznot kell összegyűjtenie a* `Player` *ágensnek ahhoz, hogy reprodukálódhasson. A reprodukció nyomán épp ennyivel fog csökkeni az addig gyűjtött összhaszna.*

- **memlimit**
  - Típusa: `int`
  - Teljesen opcionális
  - Alapértelmezett értéke (ha nincs megadva): `1000`
  - Értékkészlete: tetszőleges pozitív egész szám (legalább `1`)
  - Leírás: *arra szolgál, hogy megadjuk, hogy a játék során egyszerre legfeljebb hány különböző ellenféllel kapcsolatban tárolhasson emlékeket a* `Player` *ágens. Amennyiben elérjük ezt a limitet, és újabb ellenféllel találkozunk, akivel addig még nem lettünk összepárosítva, úgy a legrégebben frissített ellenfelünkkel kapcsolatos emlékeket töröljük az emlékezetünkből, és helyette bevesszük az új ellenféllel kapcsolatos legfrissebb emlékeket.*
- **oppmem_limit**
  - Típusa: `int`
  - Teljesen opcionális
  - Alapértelmezett értéke (ha nincs megadva): `4`
  - Értékkészlete: tetszőleges egész szám, de legalább `1`
  - Leírás: *arra szolgál, hogy megadjuk, hogy a játék során egy ellenféllel kapcsolatban legfeljebb hány emléket (megfigyelést) tárolhasson a* `Player` *ágens az emlékezetében. Amennyiben egy adott ellenfél kapcsán elérjük ezt a limitet, úgy elhagyjuk az adott ellenféllel kapcsolatos legrégebbi emléket, és helyette bevesszük a legújabbat.*

A paraméterek nevénél megfigyelhettük, hogy a JADE-es ágensekkel ellenben itt most valódi argumentum-neveket soroltunk fel. Arról van szó, hogy amíg a JADE-es ágensek esetében a paraméterek „névtelenek”, és csak a sorrendjük számít, addig JADEX esetében ez pont fordítva van: `név=érték` párok formájában, ámde tetszőleges sorrendben adhatjuk meg az ágensek paramétereit.

Eclipse-ben a legkönnyebb elindítani az ágenseket. A JADE-es ágensek indításához hasonlóan elég, ha jobb gombbal rákattintunk a JADEX-es ágensnek megfelelő XML-re, és az `EJADE/Deploy Agent(s)` menüpontot választjuk. Ennek hatására feljön a szokásos ablak, ahol megadhatjuk az ágens lokális nevét, és paramétereinek értékét.

NAGYON FONTOS (!!!): Amennyiben a fentiek közül szeretnénk megadni valamelyik paramétert (mindegyik opcionális), úgy a paraméter-lista elejére oda kell írnunk a JADEX-es ágens konfigurációjának nevét (alapból `default`), és csak utána sorolhatjuk fel a paraméternév-érték párokat. Pl. `default gui=true gid=\\\"pd\\\" myType=5`

Az Eclipse-es indítás hatására lényegében a `JadeAgentAdapter` ágens hívódik meg, és ez fogja megvalósítani/interpretálni az általunk indítani kívánt ágenst. Az Eclipse-es indítás hatására tehát nagyjából a következő utasítás kerül végrehajtása.

Példa egy `Player` ágens indítására (GUI bekapcsolva, Fogolydilemma, Nash-egyensúlyi program):

```
java jade.Boot -container pa:jadex.adapter.jade.JadeAgentAdapter(msclab01.gametheory_lab.PlayerAgent.Player "default gui=true gid=\\\"pd\\\" myType=5")
```

Ebből tehát jól látszik, hogy az ágenst valójában egy JADEX-hez mellékelt JADE-es `jadex.adapter.jade.JadeAgentAdapter` ágens valósítja meg, melynek valójában 2 paramétere van: az első az ágenst leíró XML fájl elérése (`msclab01.gametheory_lab.PlayerAgent.Player`), a második pedig egy String (idézőjelek között): a konfiguráció neve, és utána a `paraméternév=érték` párok felsorolása (String-paraméterértékek megadásánál escape-karakterekre van szükség).

Amennyiben több JADEX-es ágenst is el szeretnénk indítani egyszerre, úgy célszerű a fentebbi utasítással dolgozni Eclipse-ben: `Run/Open Run Dialog…/Arguments/…` Miután egymás után felsoroltuk az ágenseket, végül egyetlen gombnyomással (`Run`) elindíthatjuk őket. Például az `Program arguments` részbe írhatjuk a következőt:

```
-container -port 1099 -host localhost
ga:msclab01.gametheory_lab.GameAgent.GameAgent(5x4x3 800 3)
pa0:jadex.adapter.jade.JadeAgentAdapter(msclab01.gametheory_lab.PlayerAgent.Player
"default gid=\\\"5x4x3\\\"")
pa1:jadex.adapter.jade.JadeAgentAdapter(msclab01.gametheory_lab.PlayerAgent.Player
"default gid=\\\"5x4x3\\\"") ua:msclab01.gametheory_lab.UserAgent.UserAgent(5x4x3)
```

…a `VM arguments` részbe pedig még indítás előtt írjuk a következőket:

```
-Xss321k -Xms32m -Xmx1024m -Dhttp.proxyHost= -Dhttp.proxyPort=
```

Indítás (`Run`) után, ha fut már a JADE platform, összesen 3 ágens fog létrejönni egyetlen új konténerben a `\jade\src\msclab01\gametheory_lab\games\5x4x3.nfg` játékot feltételezve (ami 3 szereplős, és 5*4*3=60 tiszta stratégia-kombináció/kimenetel van benne). A 3 ágensből 1 felhasználói `UserAgent` ágens (neki csak a játék nevét adjuk meg, és kiindulási hasznát nem állítjuk át az alapértelmezett `0.0`-ról), a másik 2 ágens pedig olyan gépi `Player` ágens, amelynek ugyancsak minden paramétere alapértelmezett (azaz pl. véletlenszerűen játszanak), kivéve a játszott játék nevét.

A `VM arguments` részben pedig lényegében 3 dolgot állítottunk be (annak a JVM-nek, amely az előbbi 3 ágenst tartalmazó konténert fogja futtatni):

1) `Xss321k`: az indított JVM-en belül minden Java-thread stack-jének mérete legyen **321 Kbyte**.
2) `Xms32m`: az indított JVM indítás után **32 Mbyte** memóriát allokáljon (initial heap size).
3) `Xmx1024m`: az indított JVM legfeljebb **1 Gbyte** memóriát allokálhat (maximal heap size).

4. Irodalomjegyzék

[1] Neumann, J., Morgenstern, O.: *Theory of games and economic behavior*. Princeton University Press (1947)

[2] Harsányi, J. C.: *Games with incomplete information played by Bayesian players I-II-III*, In Management Science, Vol. 14. (1967-1968), pp. 159–182, 320–334, 486–502

[3] Maynard-Smith J.: *Evolution and the Theory of Games*, Cambridge University Press (1982)

[4] Serrano, R.: *The Theory of Implementation of Social Choice Rules*, In SIAM Review, Vol. 46. (2004), pp. 377-414

[5] Nash, J. F.: *Non-cooperative games*, In. Annals of Mathematics, Vol. 54. (1951), pp. 286–295

[6] Axelrod R.: *The Evolution of Cooperation*, Basic Books (1984)

[7] Kovács, D. L.: *Natural Selection of Game Playing Agents*, In Proceedings of 16th International Conference on Soft Computing (MENDEL 2010), Brno, Czech Republic (2010), pp. 99-106

[8] Pokahr A., Braubach L., Lamersdorf W.: *Jadex: A BDI Reasoning Engine*, Multi-Agent Programming, Kluwer (2005), http://jadex-agents.informatik.uni-hamburg.de

[9] Bellifemine F., Caire G., Greenwood D.: *Developing multi-agent systems with JADE*, Wiley Series in Agent Technology (2007), http://jade.tilab.com

[10] McKelvey, R. D., McLennan, A. M., Turocy, T. L.: *Gambit: Software Tools for Game Theory*, Version 0.2010.09.01. (2010), http://www.gambit-project.org

[11] Hardin, G.: *The Tragedy of the Commons*, Science, Vol.162, No.3859 (1968), pp. 1243-1248

5. Függelék

5/A. Héja-Galamb (Hawk-Dove) játék

Adott a Héja-Galamb-játék, amelynek keretében a játékosok egy adott erőforrásért folytatott harcban vagy héjaként (agresszíven) vagy galambként (békésen) viselkedhetnek. Az erőforrás értéke *V*. A galambok békés úton, egyenlő arányban osztják el egymás közt az erőforrást, míg a héják harcolnak érte. A harc – legalább zéró – költsége *C*. Ebből a következően az alábbi esetek/kifizetések adódnak: (H – héja; G – galamb):

- Amennyiben egy *héja* típusú játékos egy *héjával* találkozik, úgy hasznuk fejenként: (V-C)/2
- Amennyiben egy *héja* típusú játékos egy *galambbal* találkozik, úgy a héja mindent visz (haszna V), a galambnak pedig nem marad semmi (haszna 0).
- Amennyiben egy *galamb* típusú játékos találkozik egy *héjával*, úgy hasznuk épp ugyanaz, mint előbb, csak fordítva (mivel a játék szimmetrikus).
- Amennyiben egy *galamb* típusú játékos egy *galambbal* találkozik, úgy hasznuk fejenként: V/2

Mindez bimátrix (avagy táblázatos) formában a következőképp fest.

		II. játékos	
		Héja	Galamb
I. játékos	Héja	$\left(\frac{V-C}{2};\frac{V-C}{2}\right)$	$(V;0)$
	Galamb	$(0;V)$	$\left(\frac{V}{2};\frac{V}{2}\right)$

Legyen $C<V$, akkor a kifizetések sorrendje: $0<\frac{V-C}{2}<\frac{V}{2}<V$. Ez éppen a **Fogolydilemmának** felel meg (lásd. 5/B). Ha viszont $C>V$, akkor a kifizetések sorrendje: $\frac{V-C}{2}<0<\frac{V}{2}<V$. Ez leginkább a **Gyáva Nyúl** játékhoz hasonlít (lásd. 5/C).

5/B. Fogolydilemma (Prisoners' Dilemma) játék

Egy súlyos bűntény kapcsán két gyanúsítottat tartóztat le a rendőrség. Mivel nem áll rendelkezésre elegendő bizonyíték a vádemeléshez, ezért elkülönítik őket egymástól és mindkettejüknek ugyanazt az ajánlatot teszik. Amennyiben az első fogoly vall és társa hallgat, akkor az előbbi büntetés nélkül elmehet, míg a másik, aki nem vallott, 10 év börtönt kap. Ha az első tagadja meg a vallomást és a második vall, akkor a másodikat fogják elengedni és az első kap 10 évet. Ha egyikük sem vall, akkor egy kisebb bűntényért fejenként 6 hónapot kapnak, ha viszont mindketten vallanak, fejenként 6 évet kapnak.

A játék szimmetriáját, és a kimenetelek hasznosságának sorrendjét szem előtt tartva, a következő bimátrix reprezentáció adható.

		II. játékos	
		Vall	Hallgat
I. játékos	Vall	(-2, -2)	(3, -3)
	Hallgat	(-3, 3)	(2, 2)

A „Vall" stratégiát versengőnek tekintjük, míg a „Hallgat"-ot kooperatívnak.

5/C. Gyáva Nyúl (Chicken) játék

Két autós száguld egymással szemben, és az veszít, aki elrántja a kormányt.[4] Nézzük a dolgot az ő szempontjukból: *„A legrosszabb eset nyilván az, ha **(1)** mindketten konfrontálunk, ekkor ugyanis összecsattanunk. Ennél jobb, ha **(2)** elrántom a kormányt, miközben a másik nem rántja el. Ekkor ugyanis életben maradok, habár a másik győz. Ennél jobb, ha **(3)** a másik is elrántja a kormányt, hiszen ekkor legalább nem arat győzelmet fölöttem. A legjobb eset pedig az, ha **(4)** a másik elrántja a kormányt, miközben én nem rántom el, és így én győzök.”*

A játék szimmetriáját, és hasznosságainak sorrendjét szem előtt tartva, a következő bimátrix reprezentáció adható.

		II. játékos	
		Vakmerő	Gyáva
I. játékos	Vakmerő	(-10, -10)	(1, -1)
	Gyáva	(-1, 1)	(0, 0)

[4] A játék a *„Chicken Game”* elnevezést a James Dean főszereplésével készült, legendás *„Haragban a Világgal (Rebel Without a Cause)”* c. filmből kapta (lásd. http://www.youtube.com/watch?v=LGUYsuYudVA).

5/D. Nemek Harca (Battle of Sexes) játék

Képzeljünk el egy házaspárt. A férj focimeccsre szeretne menni, míg a feleség operába. Viszont ennél fontosabb számukra, hogy mindketten jobban szeretnének együtt lenni, mint külön-külön. Sajnos azonban már nem tudnak egyeztetni. Dönteniük kell, hogy hová mennek: focimeccsre, vagy operába. Mit tegyenek?

A játék asszimetriáját, és hasznosságainak sorrendjét szem előtt tartva, a következő bimátrix reprezentáció adható.

		Feleség	
		Opera	Focimeccs
Férj	Opera	(1, 2)	(-1, -1)
	Focimeccs	(0, 0)	(2, 1)

Amennyiben a két különböző játékos egyéni érdekre való törekvését versengésnek, ellenkezőjét pedig kooperációnak tekintjük (azaz pl. a Férj esetében a „Focimeccs” versengés, míg a Feleség esetében ugyanez együttműködés), úgy a játék a sorok és oszlopok megfelelő átrendezésével a következő, szimmetrikus alakra hozható:

		II. játékos	
		Verseng	Kooperál
I. játékos	Verseng	(0, 0)	(2, 1)
	Kooperál	(1, 2)	(-1, -1)

5/E. Vezérürü (Leader) játék

Adott egy egyirányú út, és rajta egy kereszteződés, ahol két autós áll egymással szemben, és épp arra készül, hogy felhajtson az egyirányú útra. Így gondolkoznak: „Ha **(1)** mindketten elindulunk, akkor összeütközünk (ez a legrosszabb eset). Ennél azért egy fokkal jobb, ha **(2)** egyikőnk sem indul el, bár úgy időtlen időkig ebben a kereszteződésben rostokolnánk... Ha azonban **(3)** én várnék, és a másik indulna előbb, úgy legalább előbb-utóbb feljutnék az útra. A legjobb persze az volna, ha **(4)** én indulhatnék előbb, és a másik várna."

A játék szimmetriáját, és hasznosságainak sorrendjét szem előtt tartva, a következő bimátrix reprezentáció adható.

		II. játékos	
		Megy	Vár
I. játékos	Megy	(-1, -1)	(2, 1)
	Vár	(1, 2)	(0, 0)

Ugyanez a dilemma más megfogalmazásban (Mérő László *„Minden Másképp Egyforma"* c. könyvének 87. oldalán) a következő:. „Két szuperjólnevelt ember egymást tessékeli előre egy ajtóban. A versengés – ebben az esetben – az a stratégia, ha ragaszkodunk ahhoz, hogy a másik menjen ki először. Kooperál az, aki vállalva annak ódiumát, hogy a másik megveti udvariatlanságáért, hajlandó előre menni. A legrosszabb eset a kölcsönös versengés, mert akkor ott halnak éhen az ajtó előtt. Ennél egy fokkal jobb, ha kooperálnak, és összeütköznek az ajtóban, de legalább némi gyűrődés árán átjutnak. Ha az egyik játékos verseng (udvariaskodik), a másik kooperál (mondjuk, hogy udvariatlan), akkor mindketten simán átjutnak, de a versengő játékos valamivel jobban jár, mert plusz nyereségként amellett, hogy viszonylag gyorsan kijutott, még meg is vetheti partnerét a neveletlenségéért."

Láthatjuk, hogy Mérő féle megfogalmazásban az „Udvarias" (versengő) stratégia felel meg az eredeti megfogalmazásban a „Megy" stratégiának, stb. Nyilván egy-egy formálisan adott játéknak tetszőleges számú informális megfelelője adható.

5/F. Érmepárosítás (Matching Pennies) játék

Két játékos, akik mindketten rendelkeznek egy-egy pénzérmével. A pénzérmét egymás elől rejtve fejre, vagy írásra állítják, majd pedig egyszerre felmutatják az érméjüket. Ha mindkét érme „paritása” azonos (azaz fej-fej, vagy írás-írás), akkor az I. játékos nyer, a II. veszít. Ha a két érme különböző, akkor pedig a II. játékos nyer, és az első veszít.

A játék asszimmetriáját, és hasznosságainak sorrendjét szem előtt tartva, a következő bimátrix reprezentáció adható.

		II. játékos	
		Fej	Írás
I. játékos	Fej	(1, -1)	(-1, 1)
	Írás	(-1, 1)	(1, -1)

5/G. Közlegelők tragédiája (Tragedy of the commons)

Az eredeti történet [11] így szól: adott egy közlegelő, amely 10 tehenet tud eltartani úgy, hogy ekkor mindegyik tehén 10 liter tejet ad (adott idő alatt). Az egyik gazda ekkor gondol egyet és küld még egy tehenet a legelőre. Ekkor egy-egy tehénnek már kevesebb fű jut, ezért mindegyikük 10 helyett csak 9 liter tejet ad. Viszont az a gazda, amelyik 2 tehenet legeltet így összesen 2*9=18 liter tejhez jut. Ezt észreveszi egy másik gazda is, és ő is kiküld még egy tehenet a legelőre. Ekkor még kevesebb fű jut a teheneknek, és így már csak 8 liter tejet adnak fejenként. Viszont a 2 dezertőrnek fejenként 2*8=16 liter teje lesz. Mikor már 8 gazda tart 2 tehenet, már csak 2*(10-8)=4 liter tejet kapnak fejenként az eredeti 10-hez képest, és a 9. gazda már nem nyerne semmit egy második tehénnel. Ha pedig egy gazda úgy döntene, hogy visszavonja az egyik tehenét, rosszabbul járna. Ez(ek) tehát a 10-szereplős játék igencsak szub-optimális Nash-egyensúlya(i).

A közlegelő, mint erőforrás, addig működik optimálisan, amíg minden felhasználó betartja a közös megegyezéssel megállapított szabályokat. Azonban, ha egy szabályokat betartó szereplő döntési helyzetbe kerül, akkor számára bármely időpontban nyereségesebb a "dezertálás", mint a szabályok betartása – miközben a szabályokat betartó többi szereplő számára egyénenként csak mérsékelten (esetenként alig érzékelhetően) romlik a helyzet. Végső soron a szereplők azáltal, hogy a közvetlen érdekeiknek megfelelően cselekszenek, hosszabb távon, közvetve saját maguknak ártanak. A közlegelő csapdahelyzet legismertebb gyakorlati példája a környezetvédelem illetve környezetszennyezés.

Általában, ha összesen N játékos van, és abból K dezertőr, akkor a normál játékosok kifizetése N-K, míg a dezertőröké $2*(N-K)$. Ha $N=3$, akkor a játék a következő formát ölti.

		D			C		
D	D	0	0	0	2	2	1
	C	2	1	2	4	2	2
C	D	1	2	2	2	4	2
	C	2	2	4	3	3	3

4. ábra: Közlegelők tragédiája 3 játékos esetén

5/H. Általános Héja-Galamb játék (Generalized Hawk-Dove game)

Lehetőség van az 5/A szakaszban ismertetett 2-szereplős Héja-Galamb játék intuitív általánosítására [7]. Az általánosított játék N-szereplő esetén a következő:

- Ha mindenki Galamb, akkor a kifizetésük rendre *V/N*;
- Ha vannak Héják is, akkor a Galambok kifizetése *0*;
- Ha csak egyetlen Héja van, akkor a kifizetése *V*;
- Ha *K>1* darab Héja van, akkor a kifizetésük rendre *(V-C)/K*.

Látható, hogy az *N=2* speciális esetben éppen a 2-szereplős Héja-Galamb játékot kapjuk vissza (lásd. 5/A).

		Hawk			Dove		
Hawk	Hawk	1.0	1.0	1.0	1.5	1.5	0
	Dove	1.5	0	1.5	6.0	0	0
Dove	Hawk	0	1.5	1.5	0	6.0	0
	Dove	0	0	6.0	2.0	2.0	2.0

5. ábra: 3-szereplős általánosított Héja-Galamb játék (V=6, C=3) [7]