IGAZSÁGOS OSZTOZKODÁS

SZAKDOLGOZAT

Csillag Péter Benedek

Matematika BSc-matematikai elemző szakirány

Témavezető:

Jung Attila

Eötvös Loránd Tudomány Egyetem

Természettudományi Kar

Budapest

2024

Köszönetnyilvánítás

Mindenekelőtt szeretném megköszönni Jung Attillának, hogy minden kérdésemre válaszolt és mindig talált időt egy konzultációra. Köszönöm a közös munkát. Szeretnék még köszönetet mondani a családomnak és a barátaimnak, akik végig támogattak és segítettek a szakdolgozat megírásának rögös útján.

NYILATKOZAT

Név: Csillag Péter Benedek

ELTE Természettudományi Kar, szak: Matematika Bsc

NEPTUN azonosító: EER230

Szakdolgozat címe:
Igazságos osztozkodás

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

Budapest, 2024.06.03

Csillag Péter

a hallgató aláírása

Tartalomjegyzék

1. Absztrakt

A szakdolgozat a mechanizmustervezés egy új és izgalmas területével foglalkozik. Ez a terület az igazságos felosztás. Fő témámként két algoritmust dolgozok fel. A két algoritmus a Hannaneh Akrami, Noga Alon, Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn és Ruta Mehta által publikált EFX Allocations: Simplifications and Improvements [AAC+22], valamint a Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta és Pranabendu Misra: Improving EFX Guarantees through Rainbow Cycle Number [CGM+21] cikkekben található meg.

Elsőként definiáljuk a szükséges fogalmakat, majd ismertetünk egy algoritmust, melyet [AAC+22] felhasznál az eredményeihez. Ezek után bemutatjuk a dolgozat gerincét adó két algoritmust.

Lezárásként egy Python kódot készítünk az első algoritmushoz mind a megértés segítése mind a további tesztelés céljából.

2. Bevezetés

Tárgyak elosztása több fél között időtlen kérdés minden ember számára. Lehet szó pizza, szendvics vagy ruhák felosztásáról, mindenki igazságos részt követel. Ezt a problémát két fél között a közismert 'egyik oszt másik választ' módszer alkalmazásával meg is oldhatjuk. Mi történik azonban több osztozkodó között? A matematikusok több különböző módszert dolgoztak ki a lehetséges helyzetek megoldására. Erre jó példa a pizza felosztása kettőnél több fél esetén, mely egy a mindennapokban jól alkalmazható algoritmussal megoldható. A továbbiakban egy hasonló problémában mélyedünk el. Ez a probléma a legalább három fél közti osztozkodás törhetetlen tárgyak esetén, például ruhák felosztása több családtag között.

3. Korábbi eredmények

Az igazságos felosztás kérdésével foglalkozó elmélet elsőként Hugo Steinhaus Sur la division pragmatique című cikkében [Ste49] jelent meg 1949-ben. Valamint hozzájárult a korai fejlődéshez L. E. Dubins és E. H. Spanier How to cut a cake fairly című cikke [DS61], mely a The American Mathematical Monthlyban jelent meg 1961-ben. A téma ezek után fontos kérdésévé vált a matematikának, a pénzügynek valamint a számítástechnikának.

A területen végzett kutatást két nagy csoportba tudjuk sorolni a tárgyak oszthatósága szerint. Az első nagy rész végtelenül osztható tárgyakkal, míg a második oszthatatlan tárgyak csoportosításával foglalkozik.

Definiáljuk az osztozkodási feladatot.

3.1. Definíció. Legyen a játékosok száma N, a tárgyak halmaza M, az értékelések halmaza pedig v. Ekkor az osztozkodási feladat az M halmaz elemeinek felosztása N játékos között a v-ben található értékelések segítségével.

Elsőként szükségünk van egy eszközre mely segítégével meg tudjuk állapítani, hogy a játékosok mennyire értékelik a nekik kiosztott csomagot. Erre a feladatra használjuk az értékelésfüggvényt.

3.2. Definíció. Azokat a függvényeket, melyek a tárgyak halmazának minden részhalmazához nemnegagív értéket rendelnek, értékelésfüggvénynek nevezzük.

3.3. Definíció. Egy v_i értékelés additív, amennyiben $V_i(S) = \sum_{g \in S} v_i(g)$ teljesül bármely S halmazra.

3.4. Definíció. Egy értékelésfüggvényt monotonnak nevezünk, amennyiben $\forall A \subseteq S$ esetén $v_i(A) \leqslant v_i(S)$.

A legfontosabb eszközünk definiálása után tekintsük át röviden a felosztások terén végzett kutatások fontosabb eredményeit, mind az oszthatatlan, mind az osztható tárgyak esetében. Ezt az oszthatatlan tárgyak esetében a [ABFRV22] survey cikk az osztható esetben pedig Király Tamás játékelmélet jegyzete [VL24] alapján tettük. A jegyzet megtalálható a következő linken: Játékelmélet jegyzet

3.1. Osztható tárgyak

A szakdolgozat témája ugyan nem ehhez az irányhoz tartozik, azonban érdemes ismertetnünk az első területen elért sikereket is. Ez az irány szintén két részre osztható annak fényében, hogy

a felosztás milyen feltételeknek kell megfeleljen. Az első irányzat az igazságos felosztás, egy P pizza példáján szemléltetve n darab játékos esetén ez azt jelenti, hogy minden játékos a $\frac{v_i(P)}{n}$ részét kapja a pizzának. A második irányzet ezt finomítja az irigység bevezetésével.

3.5. Definíció. Legyen p_i az i. játékosnak kiosztott rész $\forall i \in [N]$. Emellett legyen a $v_i(\cdot)$ a játékos értékelésfüggvénye. Ebben az esetben i irigyli a j játékost amennyiben $v_i(p_i) < v_i(p_j)$ teljesül.

3.6. Definíció. Egy felosztás irigységmentes, ha nem létezik olyan $i, j \in [N]$ játékospár, ahol i. játékos irigyli a j. játékost.

Belátható, hogy az irigységmentes felosztás biztosan igazságos is. Mivel ekkor i szerint az ő része legalább olyan jó, mint a többi játékosé.

Most tegyük fel, hogy $v_i(p_i) < \frac{v_i(P)}{n}$, vagyis nem igazságos a felosztás. Ezt át tudjuk alakítani a következő módon: $nv_i(p_i) < v_i(P)$. Mivel azonban a felosztás irigységmentes, $v_i(p_i) \geqslant v_i(p_j)$ bármely j játékosra nézve. Ez azt jelenti, hogy a többi játékos csomagjának értéke felülről becsülhető p_i értékével. Ezen felül a csomagok i szerinti értékének összege meg kell egyezzen P értékével. Ebből adódóan $nv_i(p_i) < v_i(P)$ sosem teljesülhet egy irigységmentes felosztásban. Ezzel belátjuk, hogy minden irigységmentes felosztás egyben igazságos is.

Ezek mellett a feltételek mellett ki tudjuk jelenteni, hogy a játékosok számától függetlenül létezik igazságos felosztás osztható tárgyak halmazára. Ennek kapcsán ismertetünk egy módszert, mely minden esetben képes ilyen felosztást produkálni.

A példa kedvéért dolgozzunk továbbra is pizzával. A módszer egy mozgó késsel dolgozik, ennek folyománya, hogy minden játékos egy nagy szeletet kap és nem pedig több kisebbet. Az elv egyszerű, legyen A_i az i játékos szerint a pizza értéke. Mivel a keresett felosztás igazságos $\forall i \in [N]$ játékosra kell, hogy teljesüljön a következő egyenlőtlenség $\frac{A_i}{n} \leqslant v_i(p_i)$.Ezek alapján a módszer működése a következő. A pizza közepéhez helyezzük a kés hegyét és vágunk egyszer. Az első vágás után elkezdjük forgatni a kés pengélyét az óra járásával megegyezően. A továbbiakban bármely játékos szólhat, hogy a szelet neki megfelelő. Ekkor levágjuk a szeletet és a játékosnak adjuk. Ezt ismételjük addig, még minden játékos nem kap a pizzából.

Most nézzük meg a felosztást a k. játékos szemszögéből. Lehetséges, hogy számára már nem jut megfelelő méretű szelet? Mivel az első $k-1$ esetben nem szólt bele a vágásba, tudjuk, hogy az eddig kiosztott szeletek legfeljebb $\frac{A_k}{n}$ értékűek voltak. Ha ezt tudjuk akkor a kiosztott pizza értéke a k. játékos szerint legfeljebb $(k-1)\frac{A_k}{n}$ vagyis biztosan van elég k játékosnak, hogy egy igazságos szeletet kapjon. Ezek alapján beláthatjuk, hogy a felosztás igazságos.

A bizonyítás során nagyban támaszkodtunk arra a tényre, hogy amint megfelelő egy játékosnak egy szelet, akkor szól. Amennyiben ez nem történik meg, nem biztosított az igazságos felosztás. Érdemes azonban megjegyezni, hogy minden játékosnak célja ezt a stratégiát követni, hiszen

ha egy, a neki megfelelőnél értékesebb szeletet elvisz valaki, akkor jelentős hátrányba kerül.

Az irigységmentes felosztásról osztható tárgyak esetében a dolgozatban nem írunk mélyrehatóan, főleg a bizonyítás komplexitása miatt. Azonban érdemes megemlíteni a cikket, amiben részletesen leírják a módszert. A cikket [BT95] Steven J. Brams és Alan D. Taylor publikálta 1995-ben a The American Mathematical Monthly-ban. A cikkben felhasznált módszer azonban Selfridge és Conway nevét viseli.

3.2. Oszthatatlan tárgyak

Az osztható tárgyak után térjunk át az oszthatatlanok esetére. Ezen a területen a tárgyak törhetetlenségéből adódóan az igazságosság nem minden esetben érhető el. Azt viszont megfigyelték, hogy az irigység relaxációjával lehetséges ezen a területen is mérni az adott felosztás minőségét.

3.2.1. EF1

Az első ilyen relaxáció az EF1 melynek implicit bevezetése Richard J. Lipton, Evangelos Markakis, Elchanan Mossel és Amin Saberi nevéhez köthető [LMMS04]. A formális definíció azonban Erik Budish 2011-es publikációjában [Bud11] történt csak meg. EF1 esetén egy i játékos irigykedhet j játékosra addig, ameddig j csomagjában van olyan g tárgy melynek eltávolítása után i már nem irigykedik.

3.7. Definíció. Egy felosztás EF1 tulajdonságú i szempontjából, ha $\forall$ j$\neq$i játékosra igaz, hogy $v_i(X_i) > v_i(X_j \setminus g)$, valamely $g \in X_j$ tárgyra.

Erről a kutatási területről is bemutatunk egy algoritmust, mely játékosszámtól függetlenül megad egy EF1 felosztást.

A bemutatott módszer a Round Robin algoritmus. Állítsuk sorba a játékosokat. Ebben a sorrendben válasszon minden játékos egy tárgyat az elosztani kívánt halmazból, míg el nem fogy mind. Az algoritmus egyszerű mégis minden esetben jó felosztást ad. Ennek rövid bizonyítása a következő. Az elsőként választó játékos nem irigyelhet senkit, hiszen ő kiválasztotta a legnagyobb tárgyat minden körben. Megfigyelhetjük, hogy az n. helyen választó játékosra tekinthetünk úgy , mint az elsőre, ha elvesszük az eddig kiválasztott $n-1$ tárgyat. Mivel ezek a tárgyak mind külön csomagba kerülnek, ezt meg tudjuk tenni a végleges felosztás kialakítása után és még az EF1 feltételeinek is megfelel, hiszen minden csomagból egy kijelölt tárgyat távolítunk el.

3.2.2. EFX

A szakdolgozat szempontjából legfontosabb relaxáció az EFX. Az EFX relaxáció az EF1-hez hasonló. A különbség a kiválasztott tárgyban rejlik, ugyanis míg EF1 esetén egy specifikus tárgy létezése elégséges, EFX esetén bármely tárgy eltávolítása az irigység megszűnéséhez vezet. Ezek alapján megállapítható, hogy amennyiben egy felosztás EFX tulajdonságú, akkor EF1 tulajdonságú is. A relaxációt 2016-ban vezette be a ma is használatos nevén Ioannis Caragiannis, David Kurokawa, Herve Moulin, Ariel D. Procaccia, Nisarg Shah és Junxing Wang csoportja egy konferencia során, majd 2019-ben publikálta is cikk formájában [CKM$^+$19]. Ezt megelőzően viszont már használták near fairness néven egy Laurent Gourves, Jerome Monnot és Lydia Tlilane által írt cikkben 2014-ben [GMT14].

3.8. Definíció. Egy felosztás az i. játékos szempontjából EFX tulajdonságú, ha $\forall$ j$\neq$i játékosra igaz, hogy $v_i(X_i)>v_i(X_j \setminus g)$, $\forall\ g \in X_j$ tárgyra.

A szakdolgozat, valamint a [AAC$^+$22] cikk alapján készített algoritmus szempontjából legfontosabb eredmény a PR-algoritmus. Mivel a továbbiakban sokban építünk erre az algoritmusra, ezt részletesebben taglaljuk és egy python kódot is írunk rá. Az algoritmus Plaut és Roughgarden után kapta a nevét, ugyanis az ő cikkükben [PR20] jelent meg. Ebben a cikkben, többek között, azt az esetet vizsgálták, mikor minden játékos értékelésfüggvénye megegyezik. Az egyetlen feltétel erre a függvényre a monotonitás. Itt érdemes kihangsúlyozni, hogy az algoritmust ugyan három illetve két játékos esetére használjuk a továbbiakban, valójában nem létezik megkötés a játékosok számára vonatkozóan.

Az algoritmus egy tetszőleges X felosztásból indul és apróbb változtatásokkal eléri, hogy az új felosztásban a legkisebb csomag értéke mindig nagyobb legyen mint a korábbi esetben. Most tekintsük át a mozgatást az algoritmusban. A legértéktelenebb csomag legyen az e játékos csomagja. Ha ez a felosztás nem EFX, akkor létezik olyan i,j játékospár, melyre $\exists g \in X_j$, amire $X_i < X_j \setminus g$. Ekkor ezt a tárgyat kapja meg az e játékos. Ebben az esetben a két új csomag $X_e \cup g$ és $X_j \setminus g$. Ezek közül az első biztosan értékesebb mint X_e, a másodikra a feltételt pedig a következő egyenlőtlenség lánccal tudjuk bizonyítani:

$$X_e < X_i < X_j \setminus g$$

Így beláthatjuk, hogy minden esetben nő a legrosszabb halmaz értéke. Ez azt jelenti számunkra, hogy az algoritmus sosem látogat meg egy felosztást kétszer, valamint megtalál egy Y felosztást, amely EFX tulajdonságú.

3.2.3. Kód a PR algoritmushoz

Az alábbiakban leírt kód a PR algoritmus egy lehetséges megvalósítását mutatja be pythonban. A függvény megfelelő futása érdekében elégséges lenne egy paraméter, amely megadja a felosztást amelyből indulunk. Mivel azonban ezt fel fogjuk használni az [AAC+22] cikkben bemutatott algoritmus során, ahol több különböző értékelésfüggvényt használunk, szükségünk lesz egy második paraméterre is, amivel ki tudjuk választani a felhasznált értékelést.

A kód során felhasznált segédfüggvényeket, mint például az is_efxst az A függelékben taglaljuk mélyrehatóbban, így itt ezeket nem részletezzük.

A kód kulcs része az adjust function mely egy az EFX tulajdonságot meggátló tárgyat j. játékos csomagjából az i. játékos csomagjába helyez. Ezen segéd függvény felhaszálásával és egy megfelelően konstruált ciklussal megkapjuk a PR algoritmus kódját. Első lépésként ki kell választanunk a legértéktelenebb csomagot. Amennyiben ez a csomag nem EFX tulajdonságú, egy for ciklus segítségével kikeressük a csomagokat melyek miatt ez bekövetkezik és az adjust segítségével áthelyezünk belőlük tárgyakat.

```
def adjust(X, i, j,v):
    for good in X[j].copy():
        if value(v, X[i]) < value(v, X[j] - {good}):
            X[i].add(good)
            X[j].remove(good)
            print(f'Adjusting: {i+1} <- {j+1}')
            return X
    return X

def PR(X, number_of_the_valuation_function):
    v = number_of_the_valuation_function

    if all(is_efxst(X, i + 1, v) for i in range(len(X))):
        return X
    values = [value(v, x) for x in X]
    Xe_index = values.index(min(values))
    if not is_efxst(X,  Xe_index+ 1, v):
        for j in range(len(X)):
            if j != Xe_index:
                X = adjust(X, Xe_index, j)
                if is_efxst(X, Xe_index + 1, v):
                    break
```

```
return PR(X, v)
```

3.3. Az EFX tulajdonság relaxációi

Az EFX tulajdonságú felosztás létezése minden megkötést nélkülöző esetben -ekkor nem teszünk fel az értékelésfüggvényekről semmit, sem monotonitást, sem additivitást- még mindig nyitott kérdés. Emiatt alkalmazunk egy relaxációt EFX-re is. Ebben az esetben azonban az eddigiektől eltérően nem a tárgyak elvételével szüntetjük meg az irigységet, hanem egy α szorzó bevezetésével.

3.9. Definíció. Legyen $\alpha \in (0,1]$. Egy A felosztás α-EFX, amennyiben minden $i, j \in N$ játékospárra teljesül, hogy $v_i(A_i) > \alpha \cdot v_i(A_j \backslash g)$ bármely $g \in A_j$ esetén.

A többi relaxációhoz hasonlóan erre az esetre is hozunk egy példa algoritmust, ám a módszer komplexitása miatt ezt a 6.1 fejezetben taglaljuk részletesen.

A relaxációk definiálása után érdemes mejegyezni, hogy a dolgozat során mind a felosztásokra, mind a csomagokra használni fogjuk az EFX és α-EFX kifejezéseket. Egy csomag esetében mindig egy játékosva nézve alkalmazzuk. Ekkor egy csomag EFX i játékosra nézve, ha $\forall j$ játékos esetén $v_i(X_i) > v_i(X_j \backslash g)$, $\forall\, g \in X_j$ tárgyra. Amennyiben pedig egy felosztásról beszélünk a korábban leírt definíciók alapján tesszük azt.

4. EFX felosztás három játékos esetén

A matematikában mint sok más tudományágban, a mai napig léteznek megválaszolatlan kérdések. Az egyik ilyen nyitott kérdés az EFX tulajdonságú felosztások létezése n játékos esetén. Nem tudjuk sem alátámasztani, sem cáfolni az állítást, hogy a játékosok számától függetlenül létezik EFX tulajdonságú felosztás minden megkötést nélkülöző értékelésfüggvények esetén. A dolgozat során az [AAC+22] cikkben leírtak alapján ismertetünk egy algoritmust, mely három játékos esetében garantálja nekünk a megfelelő felosztást. A cikkben leírtaktól eltérően azonban minden értékelésfüggvénytől additivitást követelünk meg. Ennek fő oka, hogy egy könnyen átlátható egységes esetet tudjunk ismertetni, valamint így egy kicsit egyszerűsíthető a bizonyítás.

4.1. Tétel. *Minden* $I = \langle [3], M, \mathbf{v} \rangle$ *esetén ha* $v_i(\cdot)$ $\forall i$*-re additív akkor* $\exists X = \langle x_1, x_2, x_3 \rangle$ *EFX tulajdonságú felosztás.*

Ha mutatunk egy algoritmust, amely ezen feltételek mellett megtalálja az X felosztást, akkor a tételt tekinthetjük bebizonyítottnak. Az alábbiakban pontosan ezt fogjuk megtenni a [AAC+22] segítségével.

4.1. Az algoritmus

4.1.1. Kezdeti feltételek és nulladik lépés

Az algoritmus nulladik lépése a kezdeti X tetszőleges felosztást újraosztani. Ezt a PR algoritmussal teszi meg v_1-et felhasználva. Innentől a felosztás megfelel a következő feltételeknek:

- X_1 és X_2 EFX tulajdonságú az első játékos szempontjából.
- X_3 pedig a maradék két játékos közül legalább egyre nézve EFX tulajdonságú.

A feltételek adódnak a PR algoritmus tulajdonságaiból, valamint a tényből, hogy minden játékos szempontjából van legjobb részhalmaz, amely számára EFX tulajdonságú is. Ezek alapján a cimkézést meg tudjuk oldani a feltételeknek megfelelően.

Ezeket a feltételeket szeretnénk mindíg kielégíteni az algoritmus futtatása során. A feltételek mellett vezessünk be egy potenciál függvényt is, legyen ez $\phi(X) = \min(v_1(X_1), v_1(X_2))$. Ez a függvény nagyon fontos lesz számunkra, ugyanis egy lépés akkor lesz jó ha ezen a potenciálon növel.

4.2. Megfigyelés. *Ha a második és harmadik játékos közül valamelyik* X_1 *és* X_2 *közül EFX tulajdonságúnak talál egy csomagot, olyan felosztást találtunk amely EFX tulajdonságú.*

Bizonyítás. A bizonyítást azzal a feltevéssel kezdjük, hogy X_3 EFX tulajdonságú a második játékosra nézve. A harmadik játékos esete hasonlóan bizonyítható.

1. Az első esetben a második játékosnak megfelel az X_1 és X_2 részhalmazok közül egy. Ekkor ha a harmadik játékos kiválasztja a számára legértékesebb részhalmazt, ez után a második játékos következik, akinek még biztosan van egy megfelelő csomag és végül az első játékos. Ezzel a módszerrel ebben az esetben minden alkalommal EFX tulajdonságú elosztást kapunk.
2. A második esetben a harmadik játékos az, akire nézve az X_1 és X_2 közül valamelyik megfelelő. Ebben az esetben minden játékos egy részhalmazt biztosan megfelelőnek talál és ezek különbözőek, vagyis itt is létezik EFX tulajdonságú elosztás.

□

(Az eddig taglaltakhoz tartozó python kódrészlet a A.6 alfejezetben található meg.)

4.1.2. Első lépés

A továbbiakban feltételezzük, hogy sem a második sem a harmadik játékos nem tartja megfelelőnek X_1-et és X_2-t.

4.3. Definíció. Egy X_i csomag értékesebb mint X_j csomag az i. játékos szempontjából, amennyiben $v_i(X_i) > v_i(X_j)$. Ezt másképpen jelölhetjük $X_i >_i X_j$ módon is.

Ezt a jelölést a továbbiakban minden relációs jelre alkalmazzuk.

4.4. Definíció. A g_i jelölje a legértéktelenebb tárgyat X_3-ban az i játékos számára..

4.5. Állítás. Az $X_3 \setminus g_i$ a legértékesebb valódi részhalmaza X_3-nak az i játékos nézőpontjából.

4.6. Megfigyelés. $X_3 \setminus g_i \geqslant_i \max_i(X_1, X_2)$ *mind a második mind a harmadik játékos esetében.*

Bizonyítás. Nézzük a második játékos esetét (a harmadik játékos esete itt is hasonlóan bizonyítható).

A bizonyítás indirekt módon történik a következő feltevéssel: $X_1 >_2 X_3 \setminus g_3$. Amennyiben a feltevés teljesül, akkor az egyetlen ok, ami miatt X_1 nem EFX a második játékosra nézve, ha $\exists g \in X_2$ amire $X_2 \setminus g >_2 X_1$. Ebből pedig adódik a következő $X_2 >_2 X_2 \setminus g >_2 X_1 >_2 X_3 \setminus g_3$, vagyis X_2 értékesebb a második játékos szemében, mint X_3 bármely valós részhalmaza, valamint X_1. Ez azt jelenti, hogy X_2 EFX tulajdonságú a második játékos szempontjából, amit viszont a korábbi kikötés nem enged meg. Így ellentmondásra jutottunk és a megfigyelést beláttuk. □

A továbbiakban tegyük fel, hogy $X_1 <_1 X_2$, vagyis $\phi(X) = v_1(X_1)$. A 4.5 állítás és a 4.6 megfigyelés alapján az algoritmus következő lépése g_i kiválasztása X_3-ból, majd $X_1 \cup g_i$ létrehozása. Most az algoritmus két esetet különböztet meg, annak függvényében, hogy mennyire értékes $X_1 \cup g_i$.

4.1.3. Első eset

Az első esetben $X_1 \cup g_i <_i X_3 \setminus g_i$ egyenlőtlenség teljesül. Dolgozzunk most az $i = 2$ feltevéssel(az $i = 3$ eset hasonló módon bizonyítható). Itt beláthatjuk, hogy $X_3 \setminus g_2$ a második játékosra nézve EFX tulajdonságú. Ez adódik a 4.6 megfigyelésből, hiszen $X_2 <_2 X_3 \setminus g_2$ és $X_1 <_2 X_3 \setminus g_2$. Ha $X_1 \cup g_2 <_2 X_3 \setminus g_2$ teljesül $X_3 \setminus g_2$ a legértékesebb csomag a második játékos szerint a $(X_1 \cup g_2, X_2, X_3 \setminus g_2)$ felosztásból. Ebből kiindulva $X_2 <_2 X_3 \setminus g_2$ és $X_1 \cup g_2 <_2 X_3 \setminus g_2$, vagyis a $X_3 \setminus g_2$ csomag EFX tulajdonságú a második játékos szempontjából.

Következő lépésként hozzunk létre egy új $X^{(1)} = (X_1^{(1)}, X_2^{(1)}, X_3^{(1)})$ felosztását a tárgyaknak a következő módon:

- $X_1^{(1)}$ legyen $X_1 \cup g_2$ egy minimális részhalmaza úgy, hogy $X_1^{(1)} >_1 X_1$ és $\forall g \in X_1^{(1)} : X_1^{(1)} \setminus g \leqslant_1 X_1$.
- $X_2^{(1)} = X_2$.
- $X_3^{(1)} = (X_3 \setminus g_2) \cup ((X_1 \cup g_2) \setminus X_1^{(1)})$.

A lépés után be kell bizonyítani, hogy $\phi(X^{(1)}) > \phi(X)$ a korábban leírtak alapján. Ezt könnyen beláthatjuk, ugyanis $X_2^{(1)} = X_2$ és erről tudjuk, hogy $X_2 >_1 X_1$ mivel $\phi(X) = v_1(X_1)$ egy korábbi feltevés alapján. Most $X_1^{(1)}$ értékéről kell mondani valamit, de a definíciója alapján tudjuk, hogy $X_1^{(1)} >_1 X_1$. Ezekkel belátható, hogy a potenciál függvény biztosan növekedni fog a lépés után.

Elsőként le kell ellenőriznünk, hogy a harmadik játékos szempontjából EFX tulajdonságú-e az $X_2^{(1)}$. Ugyanis amennyiben ez igaz, találtunk egy jó felosztást és az algoritmus megállhat. Ha ez nem teljesül, a kezdeti feltételek teljesülését kell bebizonyítanunk.

Most a foglalkozzunk a kezdeti feltételekkel. $X_3^{(1)}$ EFX tulajdonságú a második járékosra nézve, mivel ez a legnagyobb értékű csomag. Vagyis ha $X_1^{(1)}$ és $X_2^{(1)}$ EFX az első játékosra, akkor a feltételek teljesülnek. Itt több esetet tudunk megkülönböztetni.

1. Az első eset legyen az ha a két halmaz közül legalább egy nem EFX az első játékosra nézve. Ebben az esetben elsőként beszéljünk $X_1^{(1)}$ és $X_2^{(1)}$ értékeléséről az első játékos szerint.

4.7. Megfigyelés. *$\forall g \in X_2^{(1)}$ esetében $X_1^{(1)} >_1 X_2^{(1)} \setminus g$. Valamint $\forall h \in X_1^{(1)}$ esetében $X_2^{(1)} >_1 X_1^{(1)} \setminus h$*

Bizonyítás. Nézzük az első állítást. $X_1^{(1)} >_1 X_1$ a definíciók alapján. Szintén tudjuk, hogy $\forall g \in X_2$ esetén $X_1 >_1 X_2 \setminus g$ mivel X_1 EFX tulajdonságú volt az első játékosra nézve. Ezek alapján valamint $X_2^{(1)}$ definíciójával a következőt tudjuk állítani $\forall g \in X_2 = X_2^{(1)}$ esetén:

$$X_1^{(1)} >_1 X_1 >_1 X_2 \setminus g = X_2^{(1)} \setminus g$$

Most bizonyítsuk a második állítást. Korábban feltettük, hogy $X_2 >_1 X_1$. Emellett $X_1 >_1 X_1' \setminus h \mid \forall h \in X_1^{(1)}$ adódik $X_1^{(1)}$ definíciójából. Mindent figyelembevéve a következő egyenlőtlenséget kapjuk $\forall h \in X_1^{(1)}$ esetén:

$$X_2 = X_2^{(1)} >_1 X_1 >_1 X_1^{(1)} \setminus h$$

Ezek segítségével a megfigyelés minden állítását bebizonyítottuk. □

A 4.7 megfigyelés alapján $X_1^{(1)}, X_2^{(1)}$ csak akkor nem EFX tulajdonságú az első játékos szempontjából nézve, ha $\exists g \in X_3^{(1)} : X_3^{(1)} \setminus g >_1 \min(X_1^{(1)}, X_2^{(1)})$. Ebben az esetben futtatni kell a PR algoritmust $v_1(\cdot)$ értékelés függvény felhasználásával újra. Ennek az outputja legyen $Y = (Y_1, Y_2, Y_3)$.

Most be kell látni, hogy$\phi(Y) > \phi(X)$. Ez a következő módon történik:

$$\begin{aligned}
\phi(Y) &\geqslant \min(v_1(Y_1), v_1(Y_2), v_1(Y_3)) \\
&> \min(v_1(x_1^{(1)}), v_1(X_2^{(1)}), v_1(X_3^{(1)})) \\
&= \min(v_1(x_1^{(1)}), v_1(X_2^{(1)})) \\
&= \phi(X^{(1)}) \\
&> \phi(X)
\end{aligned}$$

mivel $v_1(X_3^{(1)}) >_1 \min(v_1(x_1^{(1)}), v_1(X_2^{(1)}))$ a feltevés következményeképpen.

Most a második játékos válassza ki a számára legértékesebb részhalmazt. Ezt nevezzük el Y_3-nak. Ezzel a 4.1.1 részben leírt feltételek teljesülnek és növeltük a potenciál függvényt is.

2. A második eset az egyszerű eset, ahol mind $X_1^{(1)}$, mind $X_2^{(1)}$ részhalmaz EFX tulajdonságú az első játékosra nézve, vagyis $X^{(1)}$ felosztás teljesíti a kritériumokat és növeli a potenciál függvényt is.

Ezzel az első nagy esetet befejeztük. (Az első nagy eset kódja megtalálható az A.8 alfejezetben.)

4.1.4. Második eset

A második esetben a $X_3 \setminus g_i <_i X_1 \cup g_i$ egyenlőtlenség igaz az első lépés után létre hozott felosztásra, ahol g_i az X_3 csomagban található legértékesebb tárgy i szerint. Az első nagy esethez hasonlóan itt is a második játékos szempontjából tekintjük a felosztást. Ez azt jelenti, hogy a $X_3 \setminus g_2 <_2 X_1 \cup g_2$ egyenlőtlenség teljesül

4.8. Megfigyelés. *Ebben az esetben belátható, hogy az $X_1 \cup g_i$ a legértékesebb csomag a második és a harmadik játékos szempontjából.*

Bizonyítás. A 4.6 megfigyelést felhasználva a következő egyenlőtlenség írható le mind a második mind a harmadik játékosra:

$$X_2 <_i X_3 \setminus g_i <_i X_1 \cup g_i$$

□

Most futtassuk a PR algoritmust az $(X_3 \setminus g_2, X_1 \cup g_2)$ felosztásra a v_2 értékelés függvénnyel Ennek az eredménye legyen (Y_2, Y_3) felosztás. Most válasszon a harmadik játékos elsőnek. A korábbiak alapján feltételezhetjük, hogy Y_2 és Y_3 közül választ valamit. Legyen a választott halmaz az Y_3. Most vegyük a következő $X' = (X_2, Y_2, Y_3)$ kiosztást:

- Az első játékos kapja X_2-t
- A második játékos kapja Y_2-t
- A harmadik pedig kapja Y_3-at

Az új felosztás meghatározása után le kell ellenőriznünk, hogy megfelel-e a 4.1.1 részben megadott feltételeknek és növeli-e a potenciál függvényt. Elsőként nézzük meg az irigységeket ebben a felosztásban.

4.9. Megfigyelés. *Y_2 EFX-tulajdonságú a második, Y_3 pedig a harmadik játékosra nézve.*

Bizonyítás. Mivel (Y_2, Y_3) a PR algoritmus végeredménye v_2-re:

$$\forall h \in Y_3 : Y_2 >_2 Y_3 \setminus h$$

$$Y_2 \geqslant_2 min_2(X_3 \setminus g_i, X_1 \cup g_i) >_2 X_2$$

Ebben az esetben az első egyenlőtlenség a PR-algoritmus tulajdonságaiból adódik. A második pedig a következő egyenlőtlenségből:

$$X_2 <_2 X_3 \setminus g_2 <_2 X_1 \cup g_2$$

Ezzel beláttuk, hogy Y_2 EFX-tulajdonságú a második játékosra nézve.

Most bizonyítsuk be, hogy az Y_3 EFX tulajdonságú a harmadik játékosra nézve. Fontos belátni, hogy $Y_3 = max_3(Y_2, Y_3)$, hiszen a harmadik játékos választ elsőként. Továbbá, mivel minden értékelésfüggvény additív és (Y_2, Y_3) egy valós felosztása a $X_1 \cup X_3$ részhalmaznak, beláthatjuk a következő egyenlőtlenséget is:

$$Y_3 = max_3(Y_2, Y_3) >_3 X_2$$

Ebből adódóan

$$Y_3 \geqslant_3 max_3(Y_2, X_2)$$

Vagyis beláttuk, hogy Y_3 EFX-tulajdonságú a harmadik játékosra nézve. □

Most válasszuk szét az első játékos irigysége szempontjából az eseteket. Három lehetséges helyzet létezik.

1. A X_2 EFX-tulajdonságú az első játékosra nézve.
2. Az első játékos X_2-re vett EFX tulajdonságát mind a két másik játékos meggátolja.
3. Az első játékos X_2-re vett EFX tulajdonságát pontosan egy másik játékos gátolja.

Az első esetben találtunk egy EFX-tulajdonságú felosztást vagyis az algoritmus talált egy jó felosztást és ezzel végeztünk.

A második esetben a következő egyenlőtlenségek igazak:

$$Y_2 >_1 X_2$$

$$Y_3 >_1 X_2$$

Ebben az esetben futtassunk egy PR-algoritmust v_1 értékelésfüggvénnyel a $X^{(1)} = (X_2, Y_2, Y_3)$ felosztásra. Ennek eredménye legyen $X^{(2)}$ felosztás. Most a második játékos válassza ki a számára legjobb csomagot, amit nevezzünk el $X_3^{(2)}$-nak. Így a 4.1.1 részben leírt feltételek teljesülnek ugyanis $X_1^{(2)}, X_2^{(2)}$ az első, míg $X_3^{(2)}$ a második játékosra nézve EFX-tulajdonságú. Emellett $\phi(X^{(2)}) > \phi(X)$, mivel

$$min_1(X_1^{(2)}, X_2^{(2)}, X_3^{(2)}) >_1 min_1(X_2, Y_2, Y_3) = X_2 >_1 X_1 = \phi(X)$$

Az első egyenlőtlenség a PR-algoritmus tulajdonságaiból adódik.

A harmadik esetben feltehetjük, hogy az első játékos a második játékost irigyli.(A bizonyítás ebben az esetben is szimmetrikus.) Legyen $Y_2^{(1)}$ az Y_2 egy minimális részhalmaza, amit az első játékos jobban értékel, mint X_2-t. Változtassunk $X^{(2)}$ felosztáson a következőképpen:

- $X_1^{(2)} = X_2$
- $X_2^{(2)} = Y_2^{(1)}$
- $X_3^{(2)} = Y_3 \cup (Y_2 \setminus Y_2^{(1)})$

Most nézzük ebben a javított elosztásban a játékosok irigységét. Beláthatjuk, hogy $X_3^{(2)}$ EFX-tulajdonságú a harmadik játékosra nézve, mivel $X^{(1)}$ felosztásban ez teljesült és most $X_1^{(2)}$ vál-

tozatlan, $X_2^{(2)}$ vesztett az értékéből $X^{(2)}$-ben valamint $X_3^{(1)} \subset X_3^{(2)}$. Szintén figyeljük meg, hogy

$$\phi(X^{(2)}) = min_1(v_1(X_1^{(2)}), v_1(X_2^{(2)})) = min_1(v_1(X_2), v_1(Y_2^{(1)})) = v_1(X_2) >_1 v_1(X_1) = \phi(X).$$

Ha $X_1^{(2)}$ és $X_2^{(2)}$ EFX-tulajdonságú az első játékosra nézve akkor, $X^{(2)}$ felosztás teljesíti a 4.1.1 részben leírt feltételeket, valamint $\phi(X^{(2)}) > \phi(X)$.

Most nézzük meg azt az esetet, amikor $X_1^{(2)}$ és $X_2^{(2)}$ közül legalább egy nem EFX-tulajdonságú az első játékosra nézve. Vegyük észre, hogy $\forall h \in X_2^{(2)}$-re teljesül $X_1^{(2)} >_1 X_2^{(2)} \setminus h$ és $X_2^{(2)} >_1 X_1^{(2)}$. Ez az $X_2^{(2)} = Y_2^{(1)}$ definíciójából adódik, mivel $X_1^{(2)} = X_2$. Ezek alapján ha a két csomag közül valamelyik nem EFX az első játékosra akkor $\exists h' \in X_3^{(2)}$ melyre $X_3^{(2)} \setminus >_1 min_1(X_1^{(2)}, X_2^{(2)})$.

Ebben az esetben futtasunk PR-algoritmust $X^{(2)} = (X_1^{(2)}, X_2^{(2)}, X_3^{(2)})$ felosztásra v_1 értékelésfüggvénnyel. Most is válasszon a második játékos elsőnek a kapott $X^{(3)}$ felosztásból. Ezt a csomagot nevezzük $X_3^{(3)}$-nak. Ekkor az új felosztás megfelel a 4.1.1 részben leírt feltételeknek valamint belátható, hogy $\phi(X^{(3)}) > \phi(X)$. Ezt az eggyenlőtlenséget a következő módon láthatjuk be:

$$\begin{aligned}
\phi(X^{(3)}) &\geqslant min(v_1(X_1^{(3)}), v_1(X_2^{(3)}), v_1(X_3^{(3)})) && (1)\\
&\geqslant min(v_1(X_1^{(2)}), v_1(X_2^{(2)}), v_1(X_3^{(2)})) && (2)\\
&= v_1(X_2) && (3)\\
&> v_1(X_1) && (4)\\
&= \phi(X) && (5)
\end{aligned}$$

Az egyes egyenlőtlenség könnyen belátható a potenciál függvény definíciója alapján, a kettes számú következik a PR-algoritmus tulajdonságaiból, a harmadik az $X^{(2)}$ felosztás definíciójából kapható meg, az utolsó két lépés pedig az $X_2 >_1 X_1$ feltevésből következik. Ezzel végeztünk az algoritmus elkészítésével.

A második nagy esethez tartozó kód és magyarázat az A.9 alfejezetben található meg.

5. Az algoritmus értékelése

Megfigyelhető, hogy nem minden esetben kaptunk EFX-tulajdonságú felosztást. Ennek ellenére mégis állíthatjuk, hogy az algoritmus rekurzív futtatása minden esetben talál egy ilyen felosztást. Ennek belátásához fontos hogy minden végkimenetel növelte a potenciál függvény értékét valamint minden nem EFX-tulajdonságú outputra teljesül két feltétel:

- Az első játékosra nézve EFX-tulajdonságú X_1 és X_2 csomagok
- A második és harmadik játékos közül legalább egyre nézve pedig EFX-tulajdonságú X_3

Ezek az algoritmus futásához szükséges feltételek, vagyis ha ezek teljesülnek az output felosztásokra, akkor tudjuk rájuk futtatni újra az algoritmust a nulladik lépés nélkül. Belátható, hogy az algoritmus ismételt futtatása egy jó megoldást ad. Ennek oka, hogy véges sok lehetséges felosztás létezik és mivel a $\phi(\cdot)$ szig. mon. növő ezért nem alakulhat ki soha ciklus a felosztások között. Ennek eredményeképpen biztosan EFX tulajdonságú felosztást fogunk kapni, ha elég alkalommal futtatjuk a függvényt.

6. EFX tulajdonság több játékos esetén

Eddig csak azt az esetet néztük meg ahol három játékos között osztjuk el a tárgyakat. A továbbiakban egy olyan algoritmust mutatunk be, mely bármekkora játékosszámra működik. Habár működik nagy játékosszámra is, az algoritmus során keletkezik egy ki nem osztott tárgyak halmaza melyet a továbbiakban P jelöl, valamint a felosztások nem teljesen EFX tulajdonságúak. Ebben az esetben $(1-\varepsilon)$EFX tulajdonságról beszélünk.

6.1. Definíció. Egy felosztás i játékos szempontjából $(1-\varepsilon)$EFX tulajdonságú, ha $\forall$ j$\neq$i játékosra igaz, hogy $v_i(X_i) > (1-\varepsilon) \cdot v_i(X_j \backslash g)$, $\forall\ g \in X_j$ tárgyra. Egy $(1-\varepsilon)$EFX tulajdonságú felosztás azonban tartalmazhat egy csomagot melynek elemeit egyik játékosnak sem osztjuk ki.

Ezen kívül definiálnunk kell két speciális irigységet, melyek az algoritmus során fontos szerepet játszanak.

6.2. Definíció. Egy i játékos eléggé irigyli S halmazt amennyiben a $v_i(X_i) < (1-\varepsilon) \cdot v_i(S)$ egyenlőtlenség teljesül.

A második speciális irigység az első definíciójára épül.

6.3. Definíció. Az i játékos erősen irigyli az S halmazt, amennyiben létezik olyan valódi részhalmaza S-nek melyet i eléggé irigyel.

Ezek mellett még egy fogalmat kell definiálnunk az algoritmus elméletének megalapozásához. Ez pedig a bajnok fogalma.

Vegyünk egy X felosztást. Feltételezzük, hogy egy vagy több játékos $(1-\varepsilon)$EFX tulajdonságát elrontja az X_s halmaz, miután hozzáadtunk egy g tárgyat a ki nem osztott tárgyak halmazából. Legyen i egy ilyen játékos, vagyis $v_i(X_i) < (1-\varepsilon)v_i(S')$ néhány $S' \subset X_s \cup g$. Legyen S_i az $X_s \cup g$ legszűkebb részhalmaza, melyre $v_i(X_i) < (1-\varepsilon)v_i(S_i)$ még teljesül. Ennek következményeképpen bármely $T \subset S_i$ halmazra $v_i(X_i) > (1-\varepsilon)v_i(T)$. A bevezető után definiáljuk formálisan a bajnok fogalmát.

6.4. Definíció. Egy i játékost, akinél $v_i(X_i) < (1-\varepsilon) \cdot v_i(S_i)$ teljesül valamely $S_i \subset X_s$ halmazra, ahol S_i egyik valódi részhalmazát sem irigyli eléggé egyetlen játékos sem, X_s bajnokának nevezünk.

6.1. A Rainbow cycle algoritmus

6.1.1. Elméleti alapok

Ebben a részben a fő hangsúlyt az algoritmus lépései helyett csupán a létezésének igazolására helyezzük. Ahhoz, hogy ezt meg tudjuk tenni fontos a felhasználandó tételek, lemmák megfi-

gyelések kimondása, vagyis ezzel kezdjük el a létezés bizonyítását. Mindenek előtt azonban be kell vezetnünk a rainbow cycle szám fogalmát. Ehhez azonban be kell vezetnünk a k-osztatú gráfokat.

6.5. Definíció. Egy gráf k-osztatú, amennyiben csúcsait k darab diszjunkt csoportba tudjuk sorolni úgy, hogy a csoportokon belül nem halad él.

Mint láthatjuk a k-osztatú gráf a páros gráf általánosítása. Most már minden szükséges eszközzel rendelkezünk, R(d) definiálásához.

6.6. Definíció. Bármely pozitív, egész d esetén a rainbow cycle number vagy R(d) a legnagyobb k, melyre létezik egy irányított k-osztatú $G(\bigcup_(i \in [k])V_i, E)$ melyre

- $|V_i| \leqslant d \; \forall i \in [k]$ esetén.
- Bármely két különböző V_i és V_j osztályra teljesül G-ben, hogy minden V_i beli csúcsba érkezik pontosan egy él egy V_j beli csúcsból.
- G-ben nem létezik irányított kör mely minden osztályon legfeljebb egyszer halad át.

Ezen definíció mellett R(1)=1. Ezt a következőképpen tudjuk belátni. Ha a gráf egy pontból áll és nincs önmagába menő éle, megfelel minden követelménynek. Az első kritérium teljesül, ugyanis egy pontból áll az egész gráf, vagyis az osztály mérete legfeljebb 1. A második kritérium szintén teljesül, de mivel csak egyetlen csúcs van, nem kell élet behúzni. Az élek hiányában pedig nem lehet kör sem a gráfban, vagyis mind a három feltétel teljesül. Most nézzük meg mi történik, ha két pontból áll a gráf. Ebben az esetben két pontosztályt kapunk és a második feltételből adódóan két él található a gráfban. Ez a két él egy kört alkot, mely minden osztályt pontosan egyszer érint. Ez pedig ellentmond a harmadik kritériumnak, vagyis R(1) nem lehet 2 vagy annál több.

Ezek után mondjuk ki a szükséges lemmákat és megfigyeléseket, melyeket felhasználunk a bizonyítás során. Elsőként azonban vezessük be az E_x gráfot.

6.7. Definíció. Egy X felosztás irigység gráfja a játékosokat csúcsként reprezentálja. Egy $i \to j$ irányított él akkor és csak akkor létezik a gráfban, amennyiben $v_i(X_i) <_i v_i(X_j)$ vagyis az i. játékos irigyli a j. játékost. Az X felosztás irigység gráfja E_X

6.1. Lemma (Envy cycle elimination)**.** *Ha egy* $(1-\varepsilon)$*EFX felosztás* E_x *gráfjában létezik kör, akkor meg tudunk állapítani egy* X' *felosztást, amely* $(1-\varepsilon)$*EFX tulajdonságú valamint* $\forall i \in [n]$*-re* $v_i(X'_i) \geqslant v_i(X_i)$*, ahol* $[n]$ *a játékosok halmazát jelzi és* $E_{x'}$ *aciklikus.*

Itt a módszer, amellyel megkapható az új felosztás, egy csere. Ekkor a kialakult kör irányításával ellenkező irányba átadjuk a csomagokat. Ezt a műveletet addig ismételjük míg az E_X aciklikussá nem válik. Az így kapott új felosztás legyen X'.

A továbbiakban a három legfontosabb lemmát fogjuk ismertetni az algoritmus szempontjából.

6.2. Lemma (U1 szabály)**.** *Vegyünk egy X* $(1-\varepsilon)$*EFX felosztást. Amennyiben van egy s forrás* E_x*-ben és egy ki nem osztott tárgy, amelyre teljesül, hogy senki nem irigyli erősen* $X_s \cup g$*-t, akkor*

$$X' = \langle X_1, X_2, ..., X_s \cup g, ..., X_n \rangle$$

egy $(1-\varepsilon)$ *EFX tulajdonságú felosztás, ahol* $\forall i \in [n]$*-re* $v_i(X'_i) \geqslant v_i(X_i)$*.*

Mivel X egy $(1-\varepsilon)$EFX felosztás, nincs olyan i,j játékospár, ahol i erősen irigyli j-t. Mivel $X_s \cup g$ csomagot nem irigyli egy játékos sem erősen, vagyis az eddigi erős irigység mentesség fennmarad vagyis a felosztás továbbra is $(1-\varepsilon)$EFX felosztás.

Megfigyelhetjük, hogy ezt a lemmát egymás után legfeljebb m alkalommal tudjuk használni, ugyanis minden alkalommal csökken a ki nem osztott tárgyak halmaza. Mielőtt a fenmaradó két lemmát ismertetjük, be kell vezetnünk egy fontos definíciót.

6.8. Definíció. X' felosztás erősen Pareto-dominálja az adott X felosztást, akkor és csak akkor ha $\forall i \in [n]$-re $v_i(X'_i) \geqslant v_i(X_i)$ valamint $\exists i' \in [n]$ melyre $(1-\varepsilon) \cdot v_i(X'_{i'}) \geqslant v_i(X_{i'})$. Jelölés $X' >_{PD} X$

A definíció után készen állunk a maradék két lemma kimondására.

6.3. Lemma (U2 szabály)**.** *Vegyünk egy X* $(1-\varepsilon)$*EFX tulajdonságú felosztást és legyen P a ki nem osztott tárgyak halmaza. Ha van olyan játékos, aki eléggé irigyli P-t, akkor polinomiális időben meg tudunk határozni egy* X' *felosztást, mely* $(1-\varepsilon)$*EFX tulajdonságú valamint* $X' >_{PD} X$*.*

Ebben az esetben legyen i az a játékos aki eléggé irigyli a P-t. Ekkor a bajnok definíciót alkalmazva elkezdünk keresni egy minimális részhalmazt, melyet i még eléggé irigyel, viszont bármely valódi részhalmazát már senki nem irigyli elégé. Ez a minimális részhalmaza P-nek legyen S_i. Ekkor i a jelenlegi csomagja helyett kapja meg S_i legértékesebb részhalmazát, a régi csomagot pedig tegyük a ki nem osztott tárgyak közé. Ebben az esetben kimondható, hogy egyik játékos csomagja sem csökken, valamint i csomagja nőtt a PD feltételeinek megfelelően. S_i definíciójából adódóan pedig az új felosztás szintén $(1-\varepsilon)$EFX tulajdonságú felosztás lesz.

6.4. Lemma (U3 szabály)**.** *Vegyünk egy X* $(1-\varepsilon)$*EFX tulajdonságú felosztást. Ha létezik egy csoport forrás* $s_1, ..., s_l$ E_x*-ben, egy csoport ki nem osztott tárgy* $g_1, ..., g_l$ *valamint egy csoport játékos* $t_1, t_2, ..., t_l$ *úgy, hogy* $\forall i \in [n]$ t_i *elérhető* s_i*-ből* E_x*-ben és* t_i *a* $X_{s_{i+1}} \cup g_{i+1}$ *bajnoka. Ekkor polinomiális időben meg tudunk határozni egy* X' $(1-\varepsilon)$*EFX tulajdonságú felosztást, melyre* $X' >_{PD} X$

Itt a következőképpen járunk el. Mivel t_i az $X_{i+1} \cup g_{i+1}$ bajnoka létezik olyan S_{t_i} valódi részhalmaza az $X_{i+1} \cup g_{i+1}$ csomagnak melyet t_i még eléggé irigyel azonban ennek bármely részhalmazát egyik játékos sem irigyli. Ekkor a $s_i t_i$ út mentén cseréljük el a csomagokat, vagyis mindenki megkapja az általa irigyelt csomagot az út mentén, t_i pedig kapja meg S_{t_i}-t. Itt fontos belátni, hogy a 6.4 definícióból adódóan S_{t_i}-t egyik valódi részhalmazát sem irigyli senki sem. A keletkezett felesleges tárgyat pedig tegyük a ki nem osztottak közé. Ebben az esetben egyik játékos csomagja sem csökken, t_i csomagjára pedig mivel bajnok, kijelenthető, hogy $(1-\varepsilon)v_{t_i}(X'_{t_i}) \geqslant v_{t_i}(X_{t_i})$.

Most tegyük fel, hogy a továbbiakban sem U1 sem U2 nem felhasználható valamint E_x aciklikus. Erre a továbbiakban $(*)$ feltevésként hivatkozunk. Emellett a [CGM+21] cikk alapján feltesszük, hogy $\varepsilon \leq 1/2$.

A továbbiakban arra szeretnénk rámutatni, hogy amennyiben a ki nem osztott tárgyak halmaza elég nagy, felhasználhatjuk az U3 szabályt és így javíthatunk a felosztásunkon.

Első lépésként csoportosítsuk a ki nem osztott tárgyainkat érték szerint.

6.9. Definíció. Egy részleges X felosztásban g értékes tárgy ε mellett i játékosra nézve, ha $v_i(g) > \varepsilon * v_i(X_i)$.

6.10. Megfigyelés. *Vegyük g ki nem osztott tárgyat és s forrást. Ha egy játékos eléggé irigyli a $X_s \cup g$ csomagot, akkor a játékosnak értékes a g tárgy.*

Bizonyítás. Mivel s forrás E_x-ben, ezért $v_i(X_i) \geqslant v_i(X_s)$. Mivel i eléggé irigyli $X_s \cup g$ csomagot, ezért $v_i(X_i) < (1-\varepsilon) * v_i(X_s \cup g)$. Ezek alapján a következőképpen lehet átalakítani az egyenletet:

$$\begin{aligned}
v_i(X_i) &< (1-\varepsilon) \cdot v_i(X_s \cup g) \\
v_i(X_i) &< (1-\varepsilon) \cdot (v_i(X_s) + v_i(g)) \\
v_i(X_i) &< (1-\varepsilon) \cdot (v_i(X_i) + v_i(g)) \\
v_i(X_i) - (1-\varepsilon) \cdot v_i(X_i) &< (1-\varepsilon) \cdot (v_i(X_i) + v_i(g)) - (1-\varepsilon) \cdot v_i(X_i) \\
\varepsilon v_i(X_i) &< (1-\varepsilon) \cdot v_i(g) \\
\varepsilon v_i(X_i) &< v_i(g)
\end{aligned}$$

Az átalakítás végén az értékesség definícióját kaptuk meg, vagyis a megfigyelést bebizonyítottuk. □

Mivel az 6.2 lemma feltételei nem teljesülnek a (*) feltevés miatt, megfigyelhető, hogy minden s forráshoz és g tárgyhoz van legalább egy játékos aki erősen irigyli. Ebből azt a következtetést tudjuk levonni, hogy minden ki nem osztott tárgy értékes valamely játékosnak.

6.11. Definíció. Egy ki nem osztott tárgy a H_x halmazba tartozik, akkor és csak akkor, ha legalább $d+1$ játékos számára értékes.

6.12. Definíció. Egy ki nem osztott tárgy a L_x halmazba tartozik, akkor és csak akkor, ha legfeljebb d játékos számára értékes.

A d értéket később fogjuk meghatározni.

Amennyiben sem U1, sem U2 nem használható, E_x is aciklikus és P elemszáma meghaladja a

$$\frac{4n}{(\varepsilon \cdot h^{-1}(\frac{2n}{\varepsilon}))}$$

értéket akkor U3 használható. A továbbiakban ezt fogjuk bebizonyítani, hiszen ebben az esetben tudunk javítani a felosztásunkon. Az itt felhasznált $h^{-1}(\cdot)$ függvény a $h(d) = d \cdot R(d)$ függvény inverze. Ebből adódóan $h^{-1}\left(\frac{2n}{\varepsilon}\right)$ a legkisebb pozitív egész melyre $h(d) \geqslant \frac{2n}{\varepsilon}$.

6.13. Megfigyelés. *A (*) feltételezés mellett,* $|H_x| < \frac{2n}{\varepsilon \cdot d}$

Bizonyítás. $\forall g \in H_x$-re legyen η_g azoknak a játékosoknak a száma, akik értékesnek találják a g tárgyat. Ekkor a definíció alapján az η_g- összértékére a következő egyenlőtlenség mondható ki.

$$\sum_{g \in H_x} \eta_g > |H_x| \cdot d$$

Most felűlröl korlátozzuk le a szumma értékét $\frac{2n}{\varepsilon}$ értékkel.

Ezután megmutatjuk, hogy egy játékos számára legfeljebb $2/\varepsilon$ ki nem osztott tárgy lehet értékes. Nézzünk egy i játékost. A kezdeti feltételezés alapján U2 nem használható, vagyis

$$\begin{aligned}(1-\varepsilon)v_i(P) &\leqslant v_i(X_i)\\ v_i(P) &\leqslant \frac{v_i(X_i)}{1-\varepsilon}\\ v_i(P) &\leqslant 2v_i(X_i) \text{ mivel } \varepsilon \leqslant 1/2.\end{aligned}$$

Ha egy tárgy értékes i játékos számára, akkor def. alapján $v_i(g) \geqslant \varepsilon v_i(X_i)$. Ezek alapján egy játékos számára legfeljebb $\frac{2v_i(X_i)}{\varepsilon v_i(X_i)} = \frac{2}{\varepsilon}$ ki nem osztott tárgy lehet értékes.

$$\begin{aligned}|H_x| \cdot d &< \sum_{g \in H_x} \eta_g \leqslant n \cdot \frac{2}{\varepsilon}\\ |H_x| &< \frac{2n}{\varepsilon \cdot d}\end{aligned}$$

Ezzel a megfigyelést beláttuk. □

Most nézzük meg $|L_x|$ értékét. Pontosabban megmutatjuk, hogy $|L_x| < R(d)$. A továbbiakban $k := |L_x|$. Ehhez bevezetjük a csoportos bajnok gráf fogalmát. Elsőként minden a játékoshoz kijelölünk egy $s(a)$ forrást, amelyből elérhető E_x-ben. Ha a elérhető tőbb forrásból is, akkor önkényesen választhatjuk ki $s(a)$-t ezek közül. Most definiáljuk a k-osztatú $G = (\bigcup_{g \in L_x} V_g, E)$ gráfot, ahol V_g a Q_g-ben lévő játékosokat reprezentáló források másolataiból áll, Q_g pedig a g tárgyat értékesnek találó játékosok halmaza $\forall g \in L_x$ esetén.

Most beszéljünk a G éleiről. A V_g-ben lévő $(g, s(a))$ csúcsból akkor, és csak akkor megy él a V_h-ban lévő $(h, s(b))$ csúcsba, ha a az $X_{s(b)} \cup g$ bajnoka.

6.14. Megfigyelés. *Vegyük bármely $g, h \in L_x$ párt. A (*) feltevés mellett, minden csúcshoz V_h-ban tartozik legalább egy V_g-ből érkező él*

Bizonyítás. Vegyünk egy $(h, s(b)) \in V_h$ csúcsot. A (*) feltevés alapján, van olyan játékos, mely erősen irigyli $X_{s(b)} \cup g$ csomagot. Ha ez nem lenne igaz, tudnánk alkalmazni U1-et. A 6.10 megfigyelés alapján a játékos ebben az esetben értékesnek tartja a g tárgyat, vagyis a játékos Q_g-nak eleme. Legyen a az $X_{s(b)} \cup g$ bajnoka. Szintén egy korábbi megfigyelés alapján a erősen irigyli $X_{s(b)} \cup g$ csomagot. Ebből következik, hogy $a \in Q_g$. Ezeket összevonva belátható, hogy a $(g, s(a))$ csúcsból $(h, s(b))$ csúcsba vezető él létezik, vagyis beláttuk a megfigyelést. □

6.5. Lemma. *Amennyiben létezik egy C kör a fent definiált G gráfban, mely minden V_g osztályból legfeljebb egy csúcsot tartalmaz, akkor polinomiális időben meg tudunk határozni egy $(1-\varepsilon)$EFX-tulajdonságú X' felosztást melyre $X' >_{PD} X$.*

Bizonyítás. Legyen C=$(g_{i+1}, s_i) \to (g_{i+2}, s_{i+1}) \to ... \to (g_{j+1}, s_j)$ egy kör a G gráfban, mely minden osztályt legfeljebb egyszer érint. Most nézzük a $s_i, s_{i+1}, ..., s_j$ sort. Ha nem különbözik az összes forrás, akkor létezik egy folytatólagos részsorozat $s_{i'}, s_{i'+1}, ..., s_{j'}$, ahol minden különböző valamint $s_{j'+1} = s_{i'}$ és $i \leqslant i' < j' \leqslant j$ is igaz. Innentől dolgozzunk a $s_{i'}, s_{i'+1}, ..., s_{j'}$ rész szekvenciával. Tudjuk, hogy $\forall l \in [i'+1, j'+1]$ esetén létezik $(g_l, s_{l-1}) \to (g_{l+1}, s_l)$ él. Ez azt mutatja, hogy létezik t_{l-1} játékos, aki $X_{s_l} \cup g_l$ bajnoka és $s(t_{l-1}) = s(l-1)$. Ezek alapján t_{l-1} elérhető s_{l-1}-ből E_x-ben. Mivel $s_{i'}, s_{i'+1}, ..., s_{j'}$ különbözők a reprezentált játékosok is különbözők $(a_{i'}, a_{i'+1}, ..., a_{j'})$. Ezek alapján van egy csoport különböző forrásunk $s_{i'}, s_{i'+1}, ..., s_{j'}$ E_x-ben, különböző kiosztatlan tárgyaink $g_{j'+1}, g_{i'}, ..., g_{j'}$ valamint különböző játékosok $t_{i'}, t_{i'+1}, ..., t_{j'}$. Ebben az esetben U3 alkalmazható vagyis tudunk polinomiális időben megfelelő felosztást találni. □

Ezek után már kész vagyunk felső korlátot adni L_x méretére. Megfigyelhető, hogy G csúcsosztályainak száma megegyezik L_x méretével.Ezért olyan felső korlátra van szükség, mely meggátolja egy C kör létezését a G gráfban.

6.6. Lemma. *Vegyünk egy X $(1-\varepsilon)$EFX-tulajdonságú felosztást. Ha $|L_x| > R(d)$, létezik $(1-\varepsilon)$EFX-tulajdonságú felosztás X' mely $X' >_{PD} X$.*

Bizonyítás. $|L_x| = k$ és k megegyezik G osztályainak számával. Megfigyelhető, hogy minden osztály megfelel egy adott tárgyat értékesnek tartó játékosokat reprezentáló forrásoknak. L_x deliníciója alapján legfeljebb d db játékos tart értékesnek egy ide sorolt ki nem osztott tárgyat, vagyis minden osztály legfeljebb d csúcsból áll. Tudjuk még, hogy minden V_h beli csúcsba érkezik legalább egy él V_g valamely csúcsából. R(d) definíciója alapján, ha $k > R(d)$ akkor létezik C kör a G gráfban mely minden csoportot legfeljebb egyszer érint. Ha pedig létezik C kör akkor a 7.5 Lemma alapján meg tudunk határozni egy X' $(1-\varepsilon)$EFX-tulajdonságú felosztást mely Pareto-dominálja X felosztást. Ezzel a lemmát beláttuk. □

Érdemes megfigyelni, hogy ebben az esetben a polinomiális futási idő nincsen garantálva, csak a létezés. Ennek oka a C kör megtalálása. Ez ugyanis egy nagy műveletigényű probléma. A polinomiális futási idő akkor lehetséges, ha a kört polinomiális időben meg tudjuk találni a gráfban. Ezt viszont nem tudjuk minden esetben megtenni $|L_x| > R(d)$ feltétel mellett.

6.1.2. Az algoritmus elkészítése

Az algoritmust egy X üres felosztással kezdjük, melyre triviális az $(1-\varepsilon)$EFX tulajdonság belátása, hiszen egyik játékos sem birtokol mégy egyetlen tárgyat sem. Ezután az algoritmus minden iterációban elsőként a 6.1 lemma alapján aciklikussá teszi E_x gráfot. Amenyiben E_x aciklikus három lehetséges eset van:

- U1 alkalmazható.
- U2 alkalmazható.
- Sem U1, sem U2 nem alkalmazható.

Az első esetben létrehozunk egy X' felostást, mely $(1-\varepsilon)$EFX tulajdonságú valamint $\forall i \in [n]$-re $v_i(X'_i) \geqslant v_i(X_i)$. Emellett csökken a ki nem osztott tárgyak halmaza is. A második esetben szinten új X' felosztást kapunk, mely $(1-\varepsilon)$EFX tulajdonságú valamint erősen Pareto-dominálja X felosztást. A harmadik esetben ennél komplikáltabb az utolsó lépés. Mindenekelőtt azonban létrehozzuk a H_x és L_x halmazokat. A 6.13 megfigyelés alapján $|H_X| < \frac{2n}{\varepsilon \cdot d}$. Innentől azonban L_x elemszámától függően két különböző eset lehetséges:

- $|L_x| \leqslant R(d)$
- $|L_x| > R(d)$

Az első esetben nem létezik a C kör a fent definiált G gráfban, így nem tudunk javítani az aktuális X felosztáson, vagyis az algoritmus ezt adja vissza megoldásként. A második esetben meg tudjuk határozni a C kört (Ennek gyors és hatékony megtalálását a következő fejezetben taglaljuk részletesen), vagyis U3 következménye képpen meg tudunk határozni egy X' felosztást, mely $(1-\varepsilon)$EFX tulajdonságú és erősen Pareto-dominálja X felosztást.

Az algoritmus befejeztével így:

- $|H_x| < \frac{2n}{\varepsilon \cdot d}$
- $|L_x| \leqslant R(d)$

Ebből adódóan a ki nem osztott tárgyak halmazának elemszámát meg tudjuk becsülni.

$$|P| \leqslant 2 \cdot \max\left(\frac{2n}{\varepsilon \cdot d}, R(d)\right)$$

Most adjuk meg d explicit értékét. Legyen d értéke a legkisebb egész szám, melyre a $\frac{2n}{\varepsilon \cdot d} \geqslant R(d)$ egyenlőtlenség még teljesül. Ebben az esetben $d = h^{-1}(\frac{2n}{\varepsilon}$, ahol $h(d) = d \cdot R(d)$, vagyis a ki nem osztott tárgyak halmazának elemszáma legfeljebb

$$\frac{4n}{\varepsilon \cdot h^{-1}(\frac{2n}{\varepsilon})}.$$

Már csak az algoritmus végességét kell megmutatnunk, vagyis be kell látnunk, hogy az algoritmus véges sok iteráció után megáll. Ezt egy polinomiális felső határ létezésével meg tudjuk tenni.

Miután E_x aciklikussá vált a következő módszerekkel határozható meg egy új X' felosztás egy iterációban:

- U1 alkalmazása.
- U2 alkalmazása.
- Egy C kör megtalálása a G gráfban majd ez alapján U3 felhasználása.

Minden esetben igaz, hogy $\forall i \in [n]$-re $v_i(X_i') \geqslant v_i(X_i)$. Ez azt jelenti, hogy nincsen olyan játékos, akinek a csomagja vesztett volna az értékéből a tulajdonosa szerint. Emellet U1 csak m egymást követö alkalommal használható, ugyanis a ki nem osztott tárgyak halmaza minden alkalommal csökken. A Pareto-dominancia okán U2 és U3 alkalmazása után $\exists i$ játékos, akinek $v_i(X_1') \geqslant v_i(X_i)$ a definíció miatt. Mivel minden játékos értékelés függvénye korlátos $W = max_{i \in [n]} v_i(M)$ értékkel felülről és sosem csökken. Ezek alapján legfeljebb poliy(n,m,W,$\frac{1}{\varepsilon}$) olyan iterációt hajthatunk végre, amely U2-t vagy U3-at használja fel. Összegezve, legfeljebb poliy(n,m,W,$\frac{1}{\varepsilon}$) $\cdot\, m$ iteráció után az algoritmus megáll.

A fent leírtak alapján belátható a következő tétel.

6.15. Tétel. *Legyen $h(d) = d \cdot R(d)$. Ebben az esetben létezik egy X $(1-\varepsilon)$EFX tulajdonságú felosztás és egy P ki nem osztott tárgyak halmaza melyekre $|P| \leqslant \frac{4n}{\varepsilon \cdot h^{-1}(\frac{2n}{\varepsilon}}$*

7. A rainbow cycle szám korlátozásai

Ebben a szakaszban a második algoritmus által használt R(d) értékét fogjuk korlátozni. Ez az érték a legnagyobb k, amelynél egy k-osztatú G digráfban nincs olyan irányított kör, amely minden csúcsosztályt legfeljebb egyszer érint. A G gráfra igaz, hogy minden V_i csúcsosztály mérete legfeljebb d, és bármely i,j $(i \neq j)$ esetén V_i minden csúcsába vezet él V_j valamely csúcsából.

7.1. Tétel. *Amennyiben,*

$$k \cdot (1 - \frac{1}{d})^{k-1} < 1$$

teljesül, akkor $R(d) < k$. *Ebből adódóan* $R(d) \leqslant (1 + \mathrm{o}(1))d \log d$

Ezt a következö módon kapjuk meg:

$$\begin{aligned} k(1 - \frac{1}{1-d})^{k-1} < ke^{-\frac{k-1}{d}} < 1 \\ k < e^{\frac{k-1}{d}} \\ log(k) < \frac{k-1}{d} \\ \dots \\ log(dlog(d)(1+\mathrm{o}(1)) < log(d)(1+\mathrm{o}(1)) \\ log(d) + log(log(d)) + c < log(d)(1+\mathrm{o}(1)) \\ log(d) + log(log(d)) + c < log(d) + 2log(log(d)) \\ c < log(log(d)) \end{aligned}$$

Az $\mathrm{o}(1)$ helyére behelyettesítettük $\frac{2log(log(d))}{log(d)}$ értéket a műveletek közben. Az utolsó egyenlőtlenség igaz nagy d esetén így az egyenlőtlenséget beláttuk.

Bizonyítás. Tegyük fel hogy, $k \cdot (1 - \frac{1}{d})^{k-1} < 1$ teljesül. Legyen S a G gráfból véletlenszerűen kiválasztott k csúcs halmaza oly módon, hogy minden V_i osztályból egy csúcsot véletlenszerűen és egyenletesen választunk. Emellett tegyük föl, hogy a választások függetlenek egymástól. Egy v csúcsra nézve legyen E_v az az esemény, hogy v-t beválasztjuk S-be és S nem tartalmaz olyan u csúcsot, melyre létezik irányított uv él. Belátható, hogy $v \in V_i$ esetén az E_v esemény valószínűsége legfeljebb

$$\frac{1}{|V_i|} \cdot (1 - \frac{1}{d})^{k-1}.$$

Mivel $v \in S$ valószínűsége ebben az esetben $\frac{1}{|V_i|}$. Valamint a gráf tulajdonságaira alapozva létezik olyan $u_j \in V_j$ melyre $u_j v$ irányított él létezik és $u_j \in S$ valószínűsége $\frac{1}{|V_j|} \geqslant \frac{1}{d}$. Ebből adódóan annak esélye, hogy olyan csúcsot választunk V_j-ből melyből nem megy irányított él v-be, legfeljebb $1 - \frac{1}{d}$. A választások függetlenségéből adódóan megkapjuk E_v valószinűségének felső

korlátját. Vagyis annak valószínűsége, hogy létezik v melyre E_v teljesül, legfeljebb

$$\sum_{i=1}^{k} |V_i| \frac{1}{|V_i|} \cdot (1 - \frac{1}{d})^{k-1} = k \cdot (1 - \frac{1}{d})^{k-1} < 1.$$

Ezek alapján kijelenthetjük, hogy annak valószínűsége, hogy az S alapján létrehozott részgráf minden csúcsának van be éle, nem nulla. Mivel annak a valószínűsége, hogy a mintában nincs kör, kevesebb mint 1, ezért a G gráfban biztosan található a korábbi feltételeknek megfelelő kör. □

Korábban említettük, hogy az előző fejezetben bemutatott algoritmus nem fut le polinomiális időben. Ennek kiküszöbölésére egy, az eredeti cikkben fel nem használt módszert alkalmazunk. A módszer összekötés ezzel a problémával a [AAC+22] cikkben került publikálásra.

A továbbiakban ezt a módszert fogjuk megismerni.

7.2. Állítás. Legyen G egy k-osztatú digráf V_i pontosztályokkal, melyek legfeljebb d csúcsot tartalmaznak. Tegyük fel, hogy minden különböző i,j esetén minden V_i beli csúcsba érkezik él egy V_j beli csúcsból és minden V_j beli csúcsba érkezik él egy V_i beli csúcsból. Emellett tegyük fel, hogy $k \cdot \left(1 - \frac{1}{d}\right)^{k-1} < 1$ is igaz. Ebben az esetben a C rainbow cycle megtalálható a G gráfban egy determinisztikus polinomiális idejű algoritmussal.

Bizonyítás. A legfontosabb eszközünk $S = s_1, s_2, ..., s_k$ G-beli csúcshalmaz megtalálásában a feltételes várható érték lesz. Itt $s_i \in V_i$ oly módon van kiválasztva, hogy az S halmaz által a G-ben létrehozott irányított részgráf minden csúcsának pozitív be-foka legyen. Ezt úgy tudjuk megtenni, hogy egyesével választjuk ki S minden elemét és közben egy potenciálfüggvényt vezetünk be S csúcsok értékelésére. Ez a bevezetett ϕ függvény a feltételes várható értékét jelöli bekövetkező E_v események számának az eddig kiválasztott s_i csúcsok alapján. A kiválasztás első lépésénél mivel S üres halmaz a ϕ értéke legfeljebb:

$$\sum_{i=1}^{k} |V_i| \frac{1}{|V_i|} \cdot \left(1 - \frac{1}{d}\right)^{k-1} = k \cdot \left(1 - \frac{1}{d}\right)^{k-1} < 1.$$

Feltéve, hogy a $s_1, s_2, ..., s_{i-1}$ már kiválasztásra került és a feltételes várható érték még mindig kisebb mint egy, válasszuk $s_i \in V_i$-t úgy, hogy az így bővített halmazra ϕ a lehető legkisebb legyen, vagyis válasszuk azt a csúcsot melyre a feltételes várható érték növekedése minimális vagy nem is növekszik. Mivel a várható érték az összes lehetséges s_i esetén kapott érték átlaga, a potenciál függvény értéke még mindig kisebb mint egy. A szükséges várható értékek kiszámítása minden lehetséges $s_i \in V_i$ esetén, ahol $|V_i| \leqslant d$, elvégezhető hatékonyan. Így S minden elemének kiválasztása után a potenciálfüggvény megadja hány csúcsra következett be az E_v esemény. Mivel ez a szám a korábbiak alapján kisebb mint egy, ez az esemény nem következik

be a kiválasztott csúcsok által létrehozott részgráfon. Így megkaptuk a kívánt S-t és a csúcsain mindig be-szomszédra mozogva egészen addig, ameddig egy már korábban meglátogatott csúcsra nem érünk megadja nekünk a keresett rainbow cyclet. □

A bizonyítás átfogó megértésének céljából érdemes beszélni a feltételes várható érték kiszámításának módjáról. Belátható ugyanis, hogy ez a legfontosabb pontja a módszernek.

Elsőként vezessünk be egy Y_v indikátor függvényt, mely vagy nulla vagy 1 értéket vesz fel egy adott ponton. A felvett érték 1, amennyiben E_v esemény bekövetkezik és nulla máskülönben. A feltételes várható érték kiválasztott $s_1, s_2, s_3, ..., s_k$ esetében

$$\mathbb{E}(\sum_v Y_v | s_1, s_2, s_3, ..., s_k)$$

módon írható le, amely a várható érték tulajdonságainak ismeretében átalakítható

$$\sum_v \mathbb{E}(Y_v | s_1, s_2, s_3, ..., s_k)$$

módon. Mivel Y_v vagy 0, vagy 1 értéket vesz fel minden esetben a várható érték megegyezik $\mathbb{P}(Y_v = 1 | s_1, s_2, s_3, ..., s_k)$ valószínűséggel. Ez a valószínűség három lehetséges értéket vehet fel.

- Az első esetben v egy olyan osztály eleme ahonnan már választottunk valamilyen s_i csúcsot. Ebben az esetben a valószínüség nulla, mivel nem választható újra.
- A második esetben létezik $s_i v$ irányított él. Ekkor a csúcsra nézve sosem következik be az esemény, vagyis valószínűsége nulla.
- A harmadik eset, amikor v csúcsba nem vezet él az eddig kiválasztott csúcsokból. Ilyenkor E_v akkor teljesül ha a többi osztályból csak olyan csúcsokat választunk melyekből nem vezet él v-be. Ebben az esetben a valószínűség $\frac{1}{|V_v|} \cdot \prod_{j=k+1}^{|L_X|} \left(1 - \frac{|N^-(v) \cap V_j|}{|V_j|}\right)$. A képletben $N^-(v)$ a v csúcs be szomszédait jelöli, azokat a csúcsokat, akikből vezet él v csúcsba.

Ezek segítségével és a gráf ismeretével hatékonyan ki tudjuk választani a megfelelő csúcsot, melyel minimálisan növeljük a feltételes várható értéket.

Ezzel a módszerrel az algoritmusunk már képes polinomiális időben $(1 - \varepsilon)$EFX felosztást találni.

Hivatkozások

[AAC+22] Hannaneh Akrami, Noga Alon, Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, and Ruta Mehta. Efx allocations: Simplifications and improvements. *arXiv preprint arXiv:2205.07638*, 2022.

[ABFRV22] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A Voudouris. Fair division of indivisible goods: A survey. *arXiv preprint arXiv:2202.07551*, 2022.

[BT95] Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

[Bud11] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[CGM+21] Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta, and Pranabendu Misra. Improving efx guarantees through rainbow cycle number. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 310–311, 2021.

[CKM+19] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019.

[DS61] Lester E Dubins and Edwin H Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1P1):1–17, 1961.

[GMT14] Laurent Gourvès, Jérôme Monnot, and Lydia Tlilane. Near fairness in matroids. In *ECAI*, volume 14, pages 393–398, 2014.

[LMMS04] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.

[PR20] Benjamin Plaut and Tim Roughgarden. Almost envy-freeness with general valuations. *SIAM Journal on Discrete Mathematics*, 34(2):1039–1068, 2020.

[Ste49] Hugo Steinhaus. Sur la division pragmatique. *Econometrica: Journal of the Econometric Society*, pages 315–319, 1949.

[VL24] Pap Júlia Végh László, Király Tamás. Játékelmélet jegyzet. *ELTE jegyzet*, 2024.

A. Az első algoritmus python kódja

Ebben a fejezetben a negyedik fejezetben feldolgozott algoritmus python kódját fogjuk átnézni. A könnyebb megértés kedvéért a kódot blokkokra bontjuk és betükóddal látunk el, melyek alapján meg lehet határozni a lépést az algoritmus leírásában is. Mindenek előtt a segédfüggvényeket kell megépíteni. Ezek a következők:

1. value(agent, bundle)
2. is_efx(X, agent)
3. is_efxst(X, agent, v)
4. efx(X)
5. isnot_efx(X, agent, agent2)
6. min_resz(X, Y, v)
7. valasztas(Y,elso,masodik,harmadik)

A kód felhasználja a PR függvényt is, azonban azt már önálló függvényként ismertettük a 3.2.3 részben.

A.1. value

A legfontosabb segéd az algoritmus során az egyes számmal jelölt value függvény. Így magától értetődik, hogy ezzel kezdjük a magyarázatot. Ez a függvény az értékelésfüggvény megfelelője. Mivel a függvények különbözők minden játékos esetében, ezért szükségünk van egy paraméterre, mely megadja melyik játékos értékeli a csomagot. A legfontosabb pedig a csomag maga, melyet értékelni szeretnénk. Az előbbit az agent, az utóbbit pedig a bundle változóval tudjuk befolyásolni. Az értékelésfüggvények additivitásából adódóan a value kódja meglehetősen egyszerű ha jól definiáljuk az adataink szerkezetét. Az általunk használt módszerrel a következőképpen néz ki a függvény:

```
if bundle:
    return sum(values[agent][good] for good in bundle)
return 0
```

Mint látható egyetlen szumma alkotja a teljes kódot. Látható, hogy az additivitást felhasználva csak összeadjuk a játékos által a csomag tárgyaihoz rendelt értékeket.

A.2. is_efx(st)

A második és harmadik függvény működése meglehetősen hasonló, ugyanis a harmadik a második egy speciális esetére ad nekünk megoldást, ezért ezeket együtt tárgyaljuk. A második függvény két paramétert kap és bool változót ad ki válaszul annak függvényében, hogy az adott játékosra nézve EFX-e a jelenlegi felosztás. Mivel minden játékoshoz saját értékelésfüggvény tartozik, a második függvény esetében nincs szükségünk specifikációra. Ezzel szemben a harmadik névszerint is_efxst esetén az st a standard rövidítése, mely arra utal, hogy minden játékos értékelése megegyezik. Ebben az esetben a plusz v paraméter jelöli melyik játékos értekelését használjuk fel. Erre a PR algoritmushoz volt szükségünk. Mind a két függvény lényegi része egy for ciklusba ágyazott for ciklus:

```
i = agent - 1
for j in range(len(X)):
    for good in X[i]:
        if value(v, X[i]) < value(v, X[j] - {good}):
            return False
return True
```

Ezeknek a segítségével a i futóváltozó bejárja az X felosztás mind a három részét még a ”good” befutja a az adott csomag minden elemét. Az if pedig eldönti a kérdést, ugyanis ha ez a feltétel teljesül olyan csomagot találtunk, mely egy elem kivétele után is értékesebb a játékos szerint, mint a sajátja. Megfigyelhetjük, hogy ebben az esetben legrosszabb esetben minden tárgyon végigmegy a keresés, vagyis n tárgy esetén n alkalommal kell az if-nek választania. Ezt ki tudjuk kerülni ha csak minden csomag legértéktelenebb tárgyára nézzük meg a döntést. Ebben az esetben minden csomagban egy alkalommal döntünk, de a minimum megtalálása ugyanannyi időbe telik, mint lefuttatni minden tárgyra a jelenlegi döntést. Ezzel ezt a két függvényt lezárhatjuk.

Az négyes számmal ellátott segéd legfőbb szerepe a kód egyszerűségének megtartása. Ez a függvény a teljes felosztásra nézett EFX tulajdonság meghatározására szolgál, amit megtehetünk három darab is_efx egymás után fűzésével de ez sok helyet foglal és a későbbi esetleges átalakítások szempontjából nem célszerű. Ennélfogva a kódja pontosan is_efx egymás után fűzése:

```
return all(is_efx(X, i + 1) for i in range(len(X)))
```

Ebben az esetben szintén egy bool változót kapunk. Magától értetődően a visszatérő érték akkor és csak akkor lesz igaz, ha minden benne lévő függvény is igaz értéket ad.

A.3. isnot_efx

Ennél a témakörnél maradva az hatodik segédfüggvény szintén az EFX tulajdonsághoz kapcsolódik. Amennyiben létezik olyan tárgy, melyet agent2-től elvéve agent1 irigyli a csomagját, a függvény megadja ezt a tárgyat. Most nézzük a kód lépéseit:

```
if X[agent2]:
    g = min(values[agent][good] for good in X[agent2])
else:
    g = None
```

Első lépésként a függvény megnézi, hogy agent2 csomagjában van-e tárgy. Hiszen ha üres a hozzá tartozó csomag, nem létezik ilyen tárgy. Erre a döntésre nagy valószínűséggel sosem lesz szükség, hiszen két tárgy esetén minden felosztás, ahol csak egy üres részhalmaz van, EFX tulajdonságú. Mivel azonban minimális értékű tárgyat keresünk, nem engedhetünk meg üres halmazt, mert az hibához vezetne. Következő lépésként a harmadik\negyedik segédfüggvénynél bemutatott alternatív megoldást használjuk:

```
if value(agent, X[agent]) < (value(agent, X[agent2]) - g):
    return g
else:
    return None
```

Érdemes megjegyezni, hogy az itt használt g egy egész szám, nem pedig egy tárgy. Mint láthatjuk nem is teszünk vele semmit, csak levonjuk egy függvény értékéből.

A.4. min_resz

A hatodik segédfüggvény három paramétert használ. Az X és Y paraméterek csomagokat jelölnek a v pedig annak a játékosnak a száma, akinek az értékelését felhasználjuk. Ekkor a függvény egy X' minimális részhalmazt ad meg X-re nézve melyre a következők teljesülnek:

- $v_v(X') > v_v(Y)$
- $\forall g \in X'$ esetén $v_v(X' \setminus g) < v_v(Y)$

Most nézzük meg a kódot, amely létrehozza nekünk ezt a minimális részhalmazt:

```
Z=set()
    while value(v,X)>value(v,Y):
```

```
        values1 = [value(v, {x}) for x in X]
        minX=min(values1)
        if (value(v,X)-minX)<value(v,Y):
            break
        for x in X:
            if value(v,{x}) == minX:
                g=x
                X= X-{g}
                Z.add(g)
                break
        values1 = [value(v, {x}) for x in X]
        minX=min(values1)
    A=[X,Z]
    return A
```

Nulladik lépésként létrehozunk egy Z halmazt, ahova azok az elemek kerülnek melyeket kivettünk az X halmazból. Első lépésként létrehozunk egy while ciklust, ami addig fut míg a v. játékos szerint X értékesebb mint Y. Ezután létrehozunk egy listát, benne X elemeinek értékeivel, majd ebből kivesszük a legkisebb értéket. Erre azért van szükségünk, hogy a ciklust le tudjuk állítani, ha a részhalmaz már megfelel a második feltételnek is. Amennyiben ez még nem teljesül, egy for ciklus segítségével ki tudjuk keresni a tárgyat a legkisebb értékkel. Ha megtaláltuk ezt a tárgyat, akkor áthelyezzük a Z halmazba. Ezek az áthelyezett tárgyak az algoritmus során még fontos szerepet fognak betölteni.

A.5. valasztas

Az utolsó segédfüggvény a valasztas. Négy paramétert adunk meg a függvénynek. Az első az Y mely egy felosztást jelöl. Ebből a felosztásból választ a három játékos a maradék három paraméter segítségével megadott sorrendben. A kód maga ebben az esetben is inkább hosszadalmas mint bonyolult:

```
X=[set(),set(),set()]
for i in range(len(Y)):
    if value(elso, Y[i]) == max(value(elso, y)for y in Y):
        X[elso-1] = Y[i]
        Y.remove(Y[i])
        break
for i in range(len(Y)):
    if value(masodik, Y[i])== max(value(masodik, y)for y in Y):
```

```
        X[masodik-1] = Y[i]
        Y.remove(Y[i])
        break
X[harmadik-1] = Y[0]
return X
```

Nulladik lépésként itt is létrehozunk egy üres listát, melyben a felosztás csomagjait tároljuk majd. A következő lépés egy for ciklus, amely végigfut a felosztás minden csomagján. Ennek segítségével kiválasztjuk a legértékesebb részhalmazt az első játékos számára, majd ezt X megfelelő indexére helyezzük és kitöröljük az Y-ból. Ezt megismételjük a második választó esetében is, majd a fennmaradó csomagot odaadjuk a játékosnak, aki még nem választott. Mivel a python indexelése nullával kezdődik és nem eggyel, ezért szükséges a játékosok számát minden alkalommal eggyel csökkenteni, hogy a megfelelő helyre kerüljön a csomag és ne adjon hibát a függvény futtatás során.

A.6. Blokk A

Az első nagyobb részét a kódnak blokk A néven fogjuk hivatkozni a leírt elemzésben. Itt definiáljuk a már tárgyalt segédeszközeinket, majd azonnal végrehajtunk egy PR-algoritmust a kezdésnél kapott felosztásunkra, majd kiosztjuk az így kapott felosztás csomagjait. A kód során X egy lista, így a játékosoknak a megfelelően indexelt helyek felelnek meg a listán. Vagyis a python kódjában X[0] az első játékoshoz tartozó csomag. Ezen a ponton eltérünk a cikkben adott indexeléstől, de ezen felül minden pontosan úgy történik, ahogy azt leírtuk korábban. A blokk utolsó lépése egy ellenőrzés, mellyel megnézzük, hogy EFX tulajdonságú felosztást talált-e a kód. Amennyiben igen, visszaadjuk ezt a felosztást és megáll az algoritmus.

```
def PR(X, number_of_the_valuation_function):...

def is_efx(X, agent):...

def is_efxst(X, agent, v):...

def efx(X):...

def isnot_efx(X, agent, agent2):...

def value(agent, bundle):...

def min_resz(X, Y, v):...
```

```
def valasztas(Y,elso,masodik,harmadik):...

def felosztas(X, values):
    Xs = PR(X, 1)
    X = [set(), set(), set()]
    X = valasztas(Xs,2,3,1)
    if efx(X):
        return X
```

A.7. Blokk B

A második blokk egy rövid mégis nagyon fontos rész, ugyanis mint olvashattuk a 4 fejezetben, az algoritmust két fő esetre lehet osztani. A B blokk fogja nekünk megalapozni ezt, mivel itt választjuk ki a második játékos csomagjából a legértéktelenebb tárgyat a játékosok szerint. Ezen tárgyak listája g és mint X esetén a játékost most is az index határozza meg. Mivel minden játékosra meg akarjuk ezt adni, kell egy for ciklus mely végigmegy a játékosokon. Emellett a legegyszerűbb maximum választási módszert használjuk, amely egy for ciklusból és egy eldöntésből áll. A legfontosabb dolog, amelyre itt figyelnünk kell, az adatok típusa, ugyanis X[i] egy részhalmazt jelöl így nincsen benne index. Ezt ebben a nyelvben ki tudjuk küszöbölni egy egyszerű list() parancsal amely a for cikluson belül ideiglenesen beindexeli a csoport elemeit így mindet át tudjuk nézni.

```
g = [None, None, None]
for j in range(3):
    for i in range( len(X[1])):
        if g[j] is None or value(j, {g[j]}) > value(j, {list(X[1][i]}):
            g[j] = list(X[1])[i]
    X[1] = X[1] - {g[1]}
```

A.8. Blokk C

Ahogyan az előző részben már említettük az algoritmus két nagy esetre bontható. Az első eset az $v_2(X_1 \cup g_2) < v_2(X_2 \setminus g_2)$ (itt a játékosok számát használjuk az indexeik helyett). Ebben az esetben létrehozunk $X_1 \cup g_2$ halmazból egy minimális részhalmazt a min_resz() segítségével. Majd az elemeket melyeket a művelet során eltávolítottunk(Z halmaz vagy A[1]) hozzácsatoljuk a második játékos csomagjához. Amennyiben ez megtörtént, felhasználjuk a 4.7 egy következményét az if megalkotásához. A következmény leírása a **1.** bekezdésben található. A döntés

eredményétől függően futtatunk egy PR()-t és kiosztjuk az új felosztás csomagjait. A blokkot nem zárjuk le return X sorral, ugyanis a felosztás fv rekurzív meghívása megteszi ezt helyettünk.

```
if (value(2, X[0]) + value(2, {g[1]})) < value(2, X[1]):
    Xf = X[0].union({g[1]})
    A = min_resz(Xf, X[0], 1)
    X[0] = A[0]
    X[1].update(A[1])
    if max((value(1, X[1]) - value(1, {good})) for good in X[1]) > mi
        Y = PR(X, 1)
        X = valasztas(Y,2,3,1)
    return felosztas(X, values)
```

A.9. Blokk D

A második nagy esethez tartozó kód kisé hosszadalmas. Emiatt a blokkot tovább bontjuk két részletre és ezeket külön tárgyaljuk. Mivel az esetszétválasztásnál ebbe az ágba kerültünk, a következő egyenlőtlenség igaz $v_2(X_1 \cup g_2) \geqslant v_2(X_2 \setminus g_2)$. Az első fontos lépés egy PR algoritmus futtatása egy két csomagból álló felosztáson. Ezért volt szükség a korábban leírt PR() során, hogy ne függjön a játékosok számától a kód futása. Miután ez megtörtént kiosztjuk a csomagokat. Ezzel a blokk első lépését befejeztük.

```
else:
    X[0].add(g[1])
    Z = [X[0], X[1]]
    Y = PR(Z, 2)
    gy = max(value(3, Y[0]), value(3, Y[1]))
    if value(3, Y[0]) == gy:
        X = [X[2], Y[1], Y[0]]
    else:
        X = [X[2], Y[0], Y[1]]
```

A második lépés egy esetszétválasztás. Annak függvényében lépünk, hogy az első játékos hogyan értékeli a felosztást. Ezt a 4.9 megfigyelés alapján tesszük meg, ugyanis ez alapján kijelenthetjük, hogy az első játékos véleménye dönti el, hogy EFX tulajdonságó felosztást kaptunk vagy sem. Mivel a megfigyelés alapján a többi játékos meg van elégedve a saját csomagjával. Három nagy esetet különböztetünk meg.

- A felosztás EFX.
- Az első mind a két játékosra irigy.
- Az első csak egy játékosra irigy.

Az első esetben a kód visszaadja a felosztást és végeztünk.

```
if is_efx(X, 1):
    return X
```

A második esetben egy utolsó PR() futtatásra van szükségünk. Ez magával hordozza a választást is természetesen. Ebben az esetben nem return X sorral zárunk, hanem újra futtatjuk a felosztás()-t.

```
elif isnot_efx(X,1,2) and isnot_efx(X,1,3):
    Y = PR(X, 1)
    X=valasztas(Y,2,3,1)
    return felosztas(X, values)
```

A harmadik és egyben utolsó esetben egy speciális és csak erre az utolsó esetre írt segédfüggvényt használunk, mely az utolso névre hallgat.

```
def utolso(X,i):
    j=3-i
    A=min_resz(X[i],X[0],1)
    X[i]=A[0]
    X[j].update(A[1])
    return X
```

Mint látható a függvény egyszerű. Ezt a segédfüggvényt azért nem tárgyaltuk a többivel, mivel ez speciálisan erre a helyzetre lett létrehozva. A függvény két paramétert használ de ezekről majd a tényleges felhasználás során beszélünk.

```
elif isnot_efx(X, 1, 2) or isnot_efx(X,1,3):
   if isnot_efx(X,1,2):
       utolso(X,2)
   else:
       utolso(X,3)
   if is_efx(X,1)==False:
       X=PR(X,1)
```

```
            valasztas(X,2,3,1)
        return felosztas(X,values)
    return X
```

Az utolsó esetben az első játékos pontosan egy másikat irigyel. Itt meg kell néznünk, melyik az. Ha megtaláltuk az irigyelt játékost, akkor futtatunk egy utolso()-t az irigyelt játékossal i paraméterként. Ilyenkor a segédfüggvény futtat egy min_resz()-t az irigyelt csomagra az első játékosra nézve. Az így eltávolított tárgyakat pedig a a nem irigyelt játékos csomagjába helyezzük. Amennyiben ez a felosztás nem megfelelő, az első játékos számára futtatnunk kell egy utolsó PR() és valasztas() kombinációt. Az esetet pedig egy rekurzív újrahívással kezdjük.