Ellenpéldák generálása számítógéppel

Szakdolgozat

Készítette: Al-Sayyed Zakariás

Matematika BSc
Alkalmazott matematikus szakirány

Témavezető: Pálvölgyi Dömötör

Docens, Matematikai Intézet, ELTE

Matematika BSc

Eötvös Loránd Tudományegyetem
Természettudományi Kar
2023

Tartalomjegyzék

Ábrák jegyzéke

Köszönetnyilvánítás

Köszönetet szeretnék mondani a témavezetőmnek, Pálvölgyi Dömötörnek a segítségéért, idejéért, és szakmai meglátásaiért, amikkel hozzájárult szakdolgozatom elkészítéséhez, illetve a családomnak a kitartó támogatásukért.

1. Bevezetés

A szakdolgozatomban azzal foglalkozok, hogy hogyan lehet különböző extremális kombinatorikai problémáknál, sejtéseknél számítógépes program segítségét felhasználni. Lineáris Programozási modelleket (továbbiakban LP) fogok írni a főbb problémákhoz, amiket a későbbiekben részletesebben leírok. Az LP-k széles körben vannak használva, elsősorban a logisztikai tervezés területén, ahol könnyen leírhatóak a folyamatok lineáris feltételekkel. Azonban ennél sokkal többféle dolog modellezhető ilyen eszközökkel, ezek közül kombinatorikai problémák lesznek a szakdolgozatban bemutatva. A megoldásukhoz egy teljesen általános, nyílt forráskódú Python könyvtárat, a pulp-ot fogom használni, egy ingyenesen elérhető fejlesztői környezetből, hogy bemutassam, nem szükséges semmilyen komolyabb technikai háttér ilyen modellek futtatásához és mennyire könnyen elérhetőek ezek az eszközök.

Az extremális kombinatorikai problémákat fogom kezdetben bevezetni, eleinte könnyebben átlátható gráf struktúráknál fogom bemutatni a lehető legtöbb él megtalálásának módszerét. Utána komplexebb struktúrákat vezetek be, a halmazrendszereket és azokon fogok különböző tulajdonságok, feltételek teljesülése mellett a lehető legtöbb élt, azaz halmazt keresni. Ezekhez már számítógép segítségét is igénybe fogom venni. Példaképp egy ismert problémát, a maximális méretű Sperner rendszer megkeresését fogom venni és bemutatom a módszernek a részleteit, amiket a későbbi modellekben alkalmazok majd. A szakdolgozatom alapötletét és a megnézendő sejtéseket Wágner Ádám Zsolt [2] cikke adta, és az alapot az LP modellekhez is ez a forrás adta, amiből feldolgozok több témát is. Többek között a 4. fejezetben Frankl [5] sejtésének, illetve az 5. fejezetben Erdős Pál kérdéséből kinőtt másik Frankltől származó sejtésnek a megcáfolásánál, illetve még szintén a 4. fejezetben Frankl egy másik sejtésében [9] levő értékeket is ellenőrizni fogok. Ezekben az esetekben a számítógépes módszereket arra lehet használni, hogy alaposabban át lehessen látni extremális halmazrendszereket, a nagy mértékű számítási kapacitás kihasználásával. Mindegyik modellnél részletezni fogom, hogy mi a kiindulási probléma, és azt hogyan lehet modellezni az eszközeimmel. A kódokat amiket használni fogok, nyilvánosan elérhetővé tettem a Githubon [1], a kódokat onnan kimásolva könnyen lehet futtattni a modelleket, mindössze a pulp könytárra van szükség. Egy modellen belül sokféle beállítási lehetőség van, az alaphalmaz méretére, időkiírásra, „warmstartra", amik rengeteg féle kódbeli konfigurációt jelentenek. Ezekre mind találni a megadott Github linken példákat, amiket minimális kódbeli változtatásokat követően bármelyiknél fel lehet használni, de a pontos részleteket, melyik kódbeli résznek mi a feladata, mindig az adott problémához tartozó fejezetben fejtek ki. Szabadon is lehet ezekkel kísérletezni, és akkor jobban

megfigyelhető milyen hatással lesz a futására, eredményére ezeknek a paramétereknek.

2. Bevezetés az extremális kombinatorikába

Az extremális kombinatorika olyan kérdésekkel foglalkozik, amik bizonyos feltételek mellett egy szélsőértéket vagy becslést adnak bizonyos, a feladatban definiált struktúra méretére. Felvezetésnek az extremális gráfelmélettel foglalkozok, hogy bemutassam milyen módszereket lehet alkalmazni szélsőértékek, konstrukciók megtalálására a gráfok témakörében. Először definiálom az alapfogalmakat, amik hozzátartoznak ennek a témának a bevezetéséhez.

2.1. Definíció. *Gráfnak nevezzük azt a $G = (V, E)$ párt, ahol V a csúcsok halmaza, $E = V \times V$ pedig az éleken egy rendezetlen pár, ami az élek két végpontját jelöli.*

2.2. Definíció. *Egy gráf akkor egyszerű gráf, ha bármely két csúcsa között legfeljebb egy él mehet és nincs benne hurokél.*

2.3. Definíció. *Egy teljes páros gráf olyan egyszerű gráf aminek a csúcsai két osztályba vannak osztva, és azok között minden él be van húzva. $K_{m,n}$ az a teljes páros gráf aminek az egyik csúcsosztálya m másik n méretű.*

Ekkor egy teljes $K_{m,n}$ páros gráfban könnyen látjuk, hogy $m \cdot n$ darab él van behúzva.

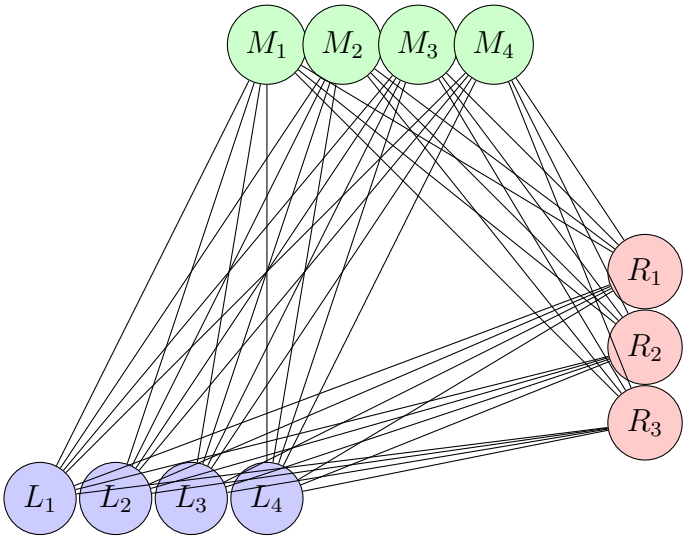
2.1. Megjegyzés. *Páros gráf esetén, a maximálisan megengedett az a két pontú teljes gráf, mert három pontú teljes részgráfot nem tartalmazhat. Ugyanis akkor menne él egy csúcsosztályon belül.*

Ezt szeretnénk a továbbiakban kiterjeszteni, mi történik akkor, amikor nem engedjük meg, hogy a gráfban egy 4 pontú teljes gráf legyen. Intuitívan arra lehet gondolni ilyenkor, hogy ha nem lehet teljes négyes, akkor próbáljuk elérni, hogy ha veszünk egy tetszőleges csúcs négyest, akkor a lehető legtöbb él legyen behúzva kötük. Próbáljuk elérni azt, hogy minden négyesből pontosan egy él hiányozzon: osszuk a csúcsokat három csoportba és a csoportok között minden élt húzzunk be. Ez tovább általánosítható, hogy ne engedjünk meg az n csúcsból álló gráfban a teljes $k+1$ csúcsú gráfot, a K_{k+1}-et. Ekkor a legtöbb élt úgy tudjuk elérni, hogy a k darab csoport méretét a lehető legkisebb különbségűre választjuk, hogy ne legyen a méretek között 1-nél nagyobb különbség.

2.4. Definíció. *$T_{n,k}$ Turán gráfnak azt a k osztályú gráfot nevezzük ami nem tartalmaz részgráfként egy $k+1$ csúcsú teljes gráfot és a k osztálya között méretben lefeljebb 1 a különbség. A gráfban pedig minden él be van húzva, kivéve amik egy adott osztályon belül mennének.*

Bizonyítás nélkül kimondom a Turán tételt.

2.1. Tétel (Turán [8])**.** *A Turán tétel azt mondja ki, hogy a $T_{n,k}$ Turán gráf azzal a tulajdonsággal bír, hogy n csúccsal rendelkezik és nincs benne $k+1$ csúcsú teljes gráf részgráfként, és ezek közül pedig a legtöbb élt tartalmazza.*



1. ábra. A $T_{11,3}$ Turán gráf

2.1. Sperner rendszer

A gráfok hasznos szemléltetőeszközök, és rengeteg feladat, folyamat modellezhető velük. Azonban ha komplexebb rendszereknél akarunk szélsőséges helyzeteket keresni, akkor nem bizonyulnak megfelelőnek, általánosabb fogalmat kell bevezetni, hogy a csúcsok közül melyek tartozhatnak egybe. Ennek feloldásaképp vezessük be a halmazrendszereket, mégpedig úgy, hogy eredetileg az E halmaz definícióját megváltoztatom az eredeti gráf definíciójában. Ehhez előbb bevezetek néhány újabb fogalmat.

2.5. Definíció. *A $P(V)$ hatványhalmaz az a halmaz ami egy V alaphalmazból képezett összes részhalmazt magába foglalja.*

Így rátérhetek a kiterjesztettebb „gráf” fogalomra.

2.6. Definíció. *Az $\mathcal{F}$ halmazcsalád egy adott V nem üres alaphalmazon vett tetszőleges részhalmaza a $P(V)$ hatványhalmaznak. Tehát az összes lehetséges halmazcsalád V nemüres alaphalmaz felett így kapható meg: $P(P(V))$*

Ez szemléletesen azt jelenti, mintha egy gráfnál olyan éleket engednénk meg, hogy egy élhez ne csak pontosan két csúcs tartozhasson, de minden élnek különböző csúcsokat kell tartalmaznia. A halmazcsaládoknál is meg lehet nézni különböző tulajdonságokat, először egy egyszerűbb tulajdonságot nézek meg, hogy legfeljebb hány halmazt (gráf értelemben „élt") lehet kiválasztani, hogy semelyik kettő ne tartalmazza egymást. Ezt formálisan is definiálom:

2.7. Definíció. *Antiláncnak nevezzük azt az* $\mathcal{F} \subset 2^{[n]}$ *halmazcsaládot amiben nincs két különböző* $A, B \in \mathcal{F}$*, hogy* $A \subset B$*.*

2.2. Lineáris Problémák megadása

Közelítsük meg ezt a problémát Lineáris Programozásbeli modellezési feladatként. Egy ilyen feladatban van egy x vektor, amiben a változók vannak, amikre lineáris feltételeket lehet megadni és egy adott célérték vektorra ezt ki lehet optimalizálni egy megoldó program, könyvtár segítségével. A modellekben minden egyes lehetséges A halmazhoz tartozni fog egy indikátor x_A változó az x vektorban. A szakdolgozat során a Python programozási nyelvet használtam PyCharm környezetben és az LP feladatok környezetéhez a pulp könyvtárat fogom használni, aminek a beimportálása alapvetően megtörténik az IDE segítségével. A következő fejezetben az adott alaphalmazon legnagyobb méretű Sperner rendszer keresése által fogom bemutatni, hogy hogyan is működik az ilyen megoldók használata.

3. Modellezés Sperner rendszer esetén

A Sperner rendszer keresése egy széles körben ismert probléma, amit az előző fejezetben elkezdtem a definiálásával, és ebben a részben tovább fogok foglalkozni vele. A megoldása is ismert, a 20. század első felében már meg lett határozva, de azért fogom bemutatni a modelljének felépítését, létrehozását, hogy példaképp lehessen egy ilyen ismert esetnél látni a módszereket, amiket alkalmazok és a későbbiekben is fel fogok használni.

A modellek során a [2] cikk volt segítségemre, megmutatta mik adják az alapját ennek a témakörnek, a modellek felépítéseinek alapját ugyanezen a példán értettem meg, de a cikkben szereplőkön felül belemegyek a kódbeli és a halmazrendszerbeli felépítésbe. A Sperner rendszerek LP modellezését és, hogy hogyan fut le a program részletesebben be fogom mutatni, mint példa feladatot, a későbbi feladatoknak nem fogok minden részén részletesen végigmenni. A fő célom az lesz, hogy meghatározzam azokat a halmazokat, amik benne lesznek a halmazcsaládban, ami az x vektorból fog látszani, és ez a lehető legtöbb halmaz legyen. Az x vektorban az elemek a lehetséges részhalmazok és x_A indikátorai lesznek annak, hogy az $A \in 2^{[n]}$ benne lesz-e a Sperner rendszerben. Ez onnan fog látszani a vektorban, hogy megkötöm az x értékeit, hogy 0 és 1 között kell, hogy legyenek és csak egész értéket vehetnek fel, 0-át, ha nincs benne és 1-et, ha igen. Azt is módosítani lehet, hogy a feladat függvényében maximalizálni vagy minimalizálni kell, ebben az esetben a maximum érték van keresve.

Ezt formálisan Lineáris Programozási feladatként így lehet felírni, aztán kódként implementálni:

$$\begin{aligned} \forall A, B \in 2^{[n]}, A \subset B : \\ x_A + x_B \leq 1 \\ max\ \underline{1}x \end{aligned}$$

```python
import pulp
# Az alaphalmazt megadom:
base_Set = "A B C D E F G H I".split()
max_Subset_Size = len(base_Set)
# Létrehozom az összes lehetséges részhalmazt az alaphalmazon:
possible_Sets = [tuple(c) for c in pulp.allcombinations(base_Set, max_Subset_Size)]
setAll = set(possible_Sets)
# Az x vektor elemei a halmazok lesznek, csak 0-1 értéket rendelünk hozzájuk
x = pulp.LpVariable.dicts( name: "set", possible_Sets, lowBound=0, upBound=1, cat=pulp.LpInteger)
set_Model = pulp.LpProblem( name: "Sperner Set System Model", pulp.LpMaximize)
set_Model += pulp.lpSum([ x[it_set] for it_set in possible_Sets])
# Ebben a lépésben határozom meg a feltételeket
for subSetA in setAll:
    for subSetB in setAll:
        subSetA = set(subSetA)
        subSetB = set(subSetB)
        if subSetA.issubset(subSetB) and subSetA != subSetB:
            subSetA = tuple(sorted(tuple(subSetA)))
            subSetB = tuple(sorted(tuple(subSetB)))
            set_Model += (pulp.lpSum([x[subSetA] + x[subSetB]]) <= 1)
# Itt fut a modell megoldása
set_Model.solve()
# Kiírom, hogy hány halmazkombináció volt összesen és aztán felsorolom őket
print(f"The choosen sets are out of a total of {len(possible_Sets)}:")
for set in possible_Sets:
    if x[set].value() == 1.0:
        print(set)
```

2. ábra. Sperner Rendszer modellezése LP-ként

A fejlesztői környezetbe, a PyCharmba le kellett töltenem a pulp könytárat ahhoz, hogy elérhetőek legyenek ennek a nyílt forráskódú LP típusú feladatokat megoldó könyvtárnak a függvényei, de mindössze ennyire volt szükség ahhoz, hogy ezt a kódot futtathassam. Ez a kód is, ahogy a bevezetésben szerepel elérhető az [1]-es hivatkozásban szereplő linken. A programkód első felében, definiáltam a problémát, beállítottam, hogy az x vektorban a halmazokhoz egész, 0 és 1 közötti érték tartozhat a határokat beleértve. Utána a modellt vettem fel, ami után az első hozzáadott rész a 11. sorban az a célfüggvény az x vektorhoz, ami jelen esetben $c = \underline{1}$ ugyanis ezzel az 1 értékek összegét akarjuk maximalizálni az x-ben. A következő lépésben a 11.-től a 20. sorig tartó részben a feltételeket fogalmazom meg. A Pythonban a típusok és beépített függvények használata érdekében szükség volt két típust is használni, tuple-t kellett az x változóihoz rendelni, de a halmazműveletek a set típuson működnek, ez a típusok közötti konverzió későbbi modellekben is jelen lesz. Minden A és B halmazpárra megnéztem, hogy $A \subsetneq B$, és minden ilyen esetben, azt várom el, hogy A és B halmaz közül csak az egyik lehet kiválasztva. Utána a kiírás maradt, hogy lehessen látni a halmazokat, amiket az LP megoldásával lehetett találni, ahol az $x_A = 1$. Ezen a szinten pedig

már futtatni is lehet ezt a modellt, hogy megoldja a pulp.

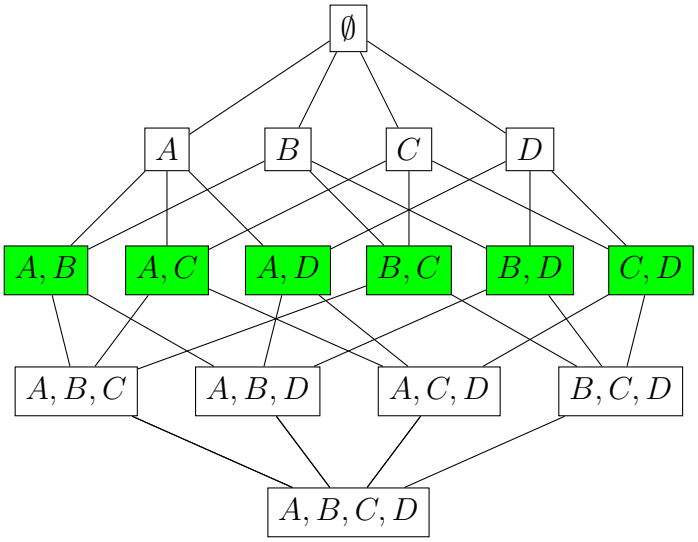
A teljes futási idő vizsgálatához néhány időlekérési hívásra meg kiírásra van szükség, a datetime nevű könyvtár használatával. A modellbeli feltételek megadásához végig kell menni két egymásba ágyazott cikluson, ami végig megy az összes $A \in \mathcal{F}$-en. A pulp rögzíti az általa a processzoron végzett számítások idejét, de csak annyiból nem lehet látni a pontos arányokat. Az időfigyeléssel ellátott futásnál kevesebb mint egy másodpercig tartott a modell felépítése, a megoldása pedig 32 másodpercig tartott. Ez nagyban tud változni az alaphalmaz méretétől meg természetesen a feladattól is.

Abban az esetben, ha megnövelem az alaphalmaz méretét 9-ről 10-re akkor a modell létrehozása $3.5 - 4$ másodperc, ami nagyjából 4-szer több idő mint előbb, ezt szemlélteti, hogy (az üres halmazt nem számolva) a 9 méretű alaphalmaznál 511 részhalmaz van, és 261121 iteráción kell összesen végigfutni, a 10 méretű alaphalmaznál 4-szer több iterációra van szükség. Az LP megoldónak is exponenciálisan több időre van szüksége, ebben az esetben 344 másodperc kellett, 11-re a modell létrehozása 7, a megoldás pedig 3877 másodperc volt. Az idő különböző heurisztikákkal akár drasztikusan gyorsítható, ha például ebben a helyzetben megszűröm a bevevendő halmazok méretét, hogy 4-től 7-ig terjedjenek akkor a megoldás 86 másodpercre csökken, ami 45-szörös gyorsulás, ha 8-ig akkor 412 ami így is több mint 9-szer gyorsabb. A heurisztikus feltételekre onnan is gondolhatunk, hogy ha túl nagy egy halmaz nem választhatunk ki olyan sokat és könnyebben tartalmaz kisebbeket, hasonlóan, ha túl kicsi akkor könnyebben tartalmazza egy nagyobb halmaz. Az optimális célérték tehát, hogy hány halmazt lehet kiválasztani, ezt szemlélteti a 3. ábra is.

3.1. LP megoldó által megadott optimumérték

A megoldó szoftver a megoldás során felhasznált lépéseket kiírja a Pythonos konzolra, így visszakövethető a megoldás menete és a közben ezeknél eltöltött idő is. A célérték szerinti maximális érték is látható, de a halmazokat külön kell kiiratni, a program végén. Nézzük meg a halmazokat amikor 9 elemű az alaphalmaz, ennek a problémának ismertebb a megoldása, de szeretném alkalmazni rá az általános hozzáállást, amit ilyen programok használatánál fogok később is használni.

Ebben az esetben megfigyelhető, hogy minden 5 méretű halmaz lett beválasztva és csak ezek, az ábrán is bejelöltem a lehető legnagyobb antiláncot zölddel, de az 4 elemű alaphalmazon van értelmezve.



3. ábra. 4 elemű alaphalmazon teljes halmazrendszer és a legnagyobb antilánc

Ahogy a 3. ábrán látni lehet azokat a halmazokat amik vonallal függőlegesen egy irányban menve elérhetőek egymásból azokat nem lehet kiválasztani, mert tartalmazzák egymást. Ez a korábban említett heurisztikát is alátámasztja, de innen látszik is, hogy $\lceil \frac{n}{2} \rceil$ vagy $\lfloor \frac{n}{2} \rfloor$ méretű részhalmazokat kell választani, hogy maximális méretű legyen a Sperner rendszer. Ekkor $\binom{n}{\lceil \frac{n}{2} \rceil}$ darab halmazt sikerül kiválasztani, aminél többet nem lehet, hogy ne tartalmazzon egymást tartalmazó halmazokat és amiket kiválasztottunk valóban teljesítik, hogy egyik sem tartalmazza a másikat, tehát antiláncot alkotnak.

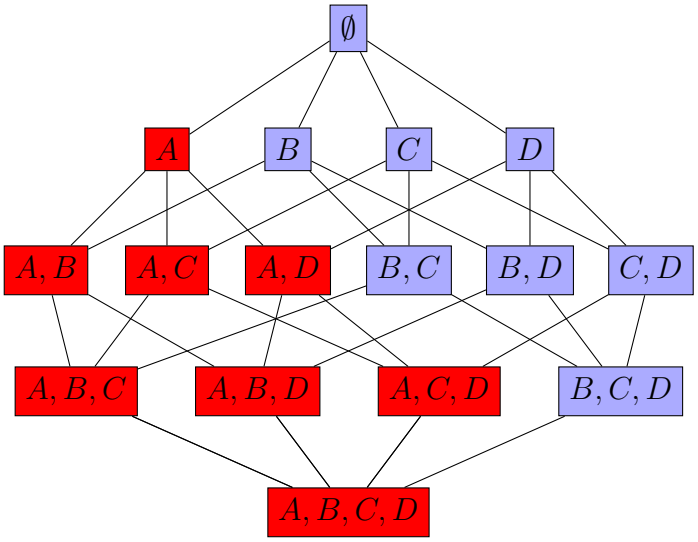
4. További extremális halmazrendszerek

A korábbi részben azt jártam körül, hogy tartalmazásra nem lehet két halmaz a halmazrendszerben, amire sikerült is választ találni. Következőként más témát fogok megvizsgálni, hogy bármely két halmaznak legyen közös eleme.

4.1. Definíció. *Metsző halmazrendszernek nevezzük $\mathcal{F}$-et ha minden $A, B \in \mathcal{F}$ halmazra teljesül, hogy $A \cap B \neq \emptyset$.*

Mielőtt konkrét ilyen halmazrendszert vennénk, először megbecsülöm a felső határát az ilyen halmazrendszerek méretének. Beosztom párba a csúcsokat: A párja $\overline{A}$ halmaz. Ekkor legfeljebb csak az egyik halmaz választható ki. 2^{n-1} halmaz és komplementere pár lesz így összesen, tehát legfeljebb ennyi halmazt lehet kiválasztani a skatulyaelv miatt, de ez még magában nem garantálja, hogy ki is lehet, ami fontos, ha egy halmazrendszer méretének maximuma van vizsgálva.

2^{n-1} méretű metsző halmazrendszer elérhető és úgy lehet a legkönnyebben szemléltetni, mintha egy csillag lenne, tehát egy fix elemet hozzáteszünk a $2^{[n-1]}$ halmaz minden eleméhez, amik $n-1$ elemű alaphalmazon vannak. Legyen az az elem amit hozzáteszünk a többi halmazhoz az A, ekkor így fog kinézni a konstrukció:



4. ábra. Az A elem által generált metsző halmazrendszer 4 elemen

Pontosan erről szól az Erdős-Ko-Rado tétel nem uniformizált változata, hogy ha $\mathcal{F} \subset 2^n$ egy metsző halmazrendszer, akkor 2^{n-1} élnél többet nem le-

het elérni, de annyit viszont igen. Bevezetem az uniformizált halmazrendszer fogalmát, hogy egy ennél általánosabb állítást is meg lehessen fogalmazni.

4.2. Definíció. *Egy halmazrendszert k-uniformnak nevezünk, ha $\forall A \in \mathcal{F}$-re teljesül, hogy A mérete pontosan k-nak kell, hogy legyen, pontosan az n elemű alaphalmaz k méretű részhalmazai.*

Következőnek pedig a k uniformizált halmazrendszerekre is kimondom az Erdős-Ko-Rado tételt:

4.1. Tétel (Erdős-Ko-Rado [3])**.** *Legyen n és k pozitív egész számok és $k \leq \frac{n}{2}$, ekkor az $\mathcal{F} \subset \binom{[n]}{k}$-ra igaz lesz, hogy $\forall A, B \in \mathcal{F}$ esetén $A \cap B \neq \emptyset$. Ekkor felülről becsülhetjük az $\mathcal{F}$ halmazrendszer méretét: $\mathcal{F} \leq \binom{n-1}{k-1}$*

Az eddigi konstrukciókat metsző halmazrendszereket könnyen meg lehetett találni, ugyanis ugyanarra a sémára épültek, hogy egy elemet fixen kiválasztottunk, hogy az legyen, amiben metszik egymást a halmazok, és annak az elemnek a fokszáma ekkor maximális lesz, tehát $\Delta(\mathcal{F}) = |\mathcal{F}|$.

Mielőtt megadnék egy sejtést a halmazrendszer mérete és a maximális fokszám között a halmazrendszer diverzitásának fogalmát vezetem be. Ezt a sejtést is a [2] cikkben találtam, a definíciókat és a témába tartozó erdeményeket is, a kódbeli megvalósítás és a speciális esetek megvizsgálása a cikktől függetlenül, általam történt.

4.3. Definíció. *Legyenek n és k pozitív egész számok, ahol $k < \frac{n}{2}$ és $\mathcal{F} \subset \binom{[n]}{k}$ egy metsző halmazcsalád. A legnagyobb fokszáma $\mathcal{F}$-nek legyen: $\Delta(\mathcal{F})$. Egy $\mathcal{F}$ halmazcsaládnak a diverzitása az a halmazcsalád méretének és a legnagyobb fokszámának a különbsége: $\rho(\mathcal{F}) = |\mathcal{F}| - \Delta(\mathcal{F})$*

Ha például az uniformizált vagy nem uniformizált Erdős-Ko-Rado tételre adott egy-egy konstrukciót nézzük, ott mindkét esetben csillagszerű volt a halmazrendszer, tehát mindkettőnél $|\mathcal{F}| = \Delta(F) \implies \rho(\mathcal{F}) = 0$, minden más esetben amikor nem csillagszerű a konstrukció, akkor legalább egy a diverzitás.

Ha $n > 6k^3$, akkor be lett bizonyítva Lemon és Palmer által [4], hogy

$$\rho(\mathcal{F}) \leq \binom{n-3}{k-2}. \tag{1}$$

Frankl ezt a szigorúbb sejtést fogalmazta meg[5]:

4.1. Sejtés (Frankl [5])**.** *Legyen $n > 3(k-1)$, és $\mathcal{F} \subset \binom{[n]}{k}$, és metsző halmazrendszer. Akkor*

$$\rho(\mathcal{F}) \leq \binom{n-3}{k-2}. \tag{2}$$

A sejtéshez az is hozzátartozik, hogy ha $n < 3(k-1)$ akkor van ellenpélda: $\mathcal{F} = \{F \in \binom{[n]}{k} : |F \cup [5]| \geq 3\}$. Erre megnézem példaképp az $n = 5,\ k = 3$ esetét, amikor $\mathcal{F} = \binom{[5]}{3}$ tehát 5 elemű alaphalmazon az összes 3 elemű részhalmazt kell venni. Így $\rho(\mathcal{F}) = 15 - 6 = 9$, tehát $\rho(\mathcal{F}) \not< \binom{2}{1}$

4.1. Modellezés LP megoldó használatához

Legegyszerűbben fix n, és k értékekre tudjuk megnézni, hogy sikerült-e ellenpéldát találni. $n = 7$, $k = 3$ esetre szeretném megkeresni a legnagyobb diverzitású $\mathcal{F}$-et.

Formálisan így lehet felírni az LP-t [2], bármely két különböző A, B halmazpárra:

$$x_A + x_B \leq 1$$

$$i \neq 1 : \sum_{A \subset [n], 1 \notin A} x_A - \sum_{A \subset [n], i \notin A} x_A \leq 0.$$

$$max \sum_{A \subset [n], 1 \notin A} x_A$$

4.1. Megjegyzés. *A szumma alsó indexében szereplő* 1*-es illetve* i *az alaphalmaz első, illetve* i*. elemére vonatkozik. Maga a teljes LP felírás a kódból is értelmezhető lehet, alatta található egy alaposabb leírás, hogy miért adja meg a keresett értéket.*

A 3. fejezetben szereplő LP modellnél egy összetettebbre lesz szükség, de sok szempontból hasonlóan fog felépülni. A kódom így áll össze:

```
1  import pulp
2  max_Subset_Size = 3
3  base_Set = "A B C D E F G".split()
4  possible_Sets = [tuple(c) for c in pulp.allcombinations(base_Set, max_Subset_Size)]
5  sizedSets = []
6  for setForSize in possible_Sets:
7      if len(setForSize) == max_Subset_Size:
8          sizedSets.append(setForSize)
9  setAll = set(sizedSets)
10
11 x = pulp.LpVariable.dicts( name: "set", sizedSets, lowBound=0, upBound=1, cat=pulp.LpInteger)
12 set_Model = pulp.LpProblem( name: "High Diversity Set System Model", pulp.LpMaximize)
13
14 optSets = []
15 for thisSet in sizedSets:
16     if set(thisSet).isdisjoint(set(possible_Sets[0])):
17         optSets.append(thisSet)
18
19 set_Model += pulp.lpSum([ x[it_set] for it_set in optSets])
20 for subSetA in setAll:
21     for subSetB in setAll:
22         subSetB = set(subSetB)
23         subSetA = set(subSetA)
24         if subSetA.isdisjoint(subSetB):
25             subSetA = tuple(sorted(tuple(subSetA)))
26             subSetB = tuple(sorted(tuple(subSetB)))
27             set_Model += (pulp.lpSum([x[subSetA] + x[subSetB]]) <= 1)
28
29 sumWOfirst = 0
30 sumWOindex = 0
31 for i in range(1, len(base_Set)):
32     for setConstr in sizedSets:
33         setConstrSet = set(setConstr)
34         if set(base_Set[0]).isdisjoint(setConstrSet):
35             sumWOfirst += x[setConstr]
36         if set(base_Set[i]).isdisjoint(setConstrSet):
37             sumWOindex += x[setConstr]
38     set_Model += ( sumWOfirst - sumWOindex ) <= 0
39     sumWOfirst = 0
40     sumWOindex = 0
41
42 set_Model.solve()
```

5. ábra. Maximális diverzitású metsző halmazrendszer modellezése LP-ként

Itt is az x vektor elemeihez halmazokat fogok rendelni, de itt még egy köztes lépésként meg kell méretezni a halmazokat, a 3 eleműek lesznek csak beadva a modellbe, ezeket a 7. sorban kezdődő ciklusban fogom kiválogatni. A 24. sorban az van garantálva, hogy metsző halmazrendszer legyen, ezt azzal a feltétellel lehet elérni, hogy

$$\forall A, B \in \binom{[n]}{k}, A \cap B = \emptyset : x_A + x_B \leq 1 \tag{3}$$

A 49. sor után már a kiírása van csak a kiválasztott halmazoknak.

Az egyik legnagyobb különbség az a 33. és 44. sor közötti feltételek, amik azért vannak hozzáadva, hogy a legnagyobb fokszámú elem az első legyen (ami a modellemben az A elem), ez megtehető csak a szimmetriát töri meg a többi elem és az A között. A másik jelentős eltérés pedig az, hogy nem

az összes 3 elemű halmaz felett fogunk optimalizálni, hanem csak azokra, amikben nincs benne az A elem, ugyanis az előző feltételek miatt az lesz a legnagyobb fokszámú elem az alaphalmazban. A lehető legtöbb halmazt akarom, hogy bekerüljön a halmazrendszerbe, ami független az A csúcstól, ugyanis úgy kapható meg a diverzitás, hogy a legnagyobb fokszámú csúcshoz megnézzük a tőle független élek számát. A további feltételek miatt egy metsző halmazrendszer is lesz a végeredmény, ami a modell futtatásával megkapható. A teljes kód elérhető a [1] források közt levő linken.

4.2. A modell futtatásának eredménye

A futási idő ilyen méretű problémánál meglehetősen gyors, néhány másodperc alatt kész is van a program. Megnéztem nagyobb halmazoknál is futást, a halmazok amiket visszaad eredményül, olyan struktúrát alkotnak, amikben nincs benne az egyik elem, viszont bármelyik másik 6 elem az pontosan 5-ször szerepel és visszaellenőrizhető, hogy metsző is a halmazrendszer.

Megnézem, hogy mennyi lesz ekkor a diverzitás a 4.1-es sejtésben:

$$\rho(\mathcal{F}) = |\mathcal{F}| - \Delta(\mathcal{F}) \leq \binom{n-3}{k-2}$$
$$\rho(\mathcal{F}) = 10 - 5 = 5 \not\leq \binom{4}{1} = 4$$

Ezzel pedig Frankl sejtését is meg lehet cáfolni a diverzitás felső becsléséről.

A következő részben Frankl egy másik sejtését fogom megvizsgálni.

4.3. Legnagyobb méretű antilánc keresése állandó átmérő mellett

A részfejezet tartalmának alapjául szintén Wágner Ádám cikke [2] szolgált. A kód, a modell és a halmazrendszer elemzését önállóan tettem hozzá, így több a cikkben nem szereplő értékre is kipróbálhattam, hogy teljesíti-e a feltételeket az eredményül kapott halmazrendszer. A Sperner rendszernél szabadon lehetett a halmazrendszert összerakni, ennél viszont bevezetem a halmazrendszer átmérőjét, mint új fogalmat.

4.4. Definíció. *Legyen $\mathcal{F} \subset 2^{[n]}$, akkor $\mathcal{F}$ átmérője, $diam(\mathcal{F})$, így adható meg:*

$diam(\mathcal{F}) = max_{A,B\in\mathcal{F}}\{|(A\backslash B) \cup (B\backslash A)|\}$.

Az átmérő így a halmazrendszerben megtalálható legnagyobb szimmetrikus differencia bármely két A és B halmazpárt véve. Másképp megfogalmazva

az, hogy mennyire tud nagy különbség lenni a halmazok között, hogy összesen hány egymástól különböző eleme lehet maximum bármely két halmaznak.

4.2. Sejtés (Frankl [5]). *Legyen n és d pozitív egész számok, és $n > d$. Legyen $\mathcal{F} \subset 2^{[n]}$ egy antilánc és $diam(\mathcal{F}) \leq d$. Akkor*

$$|\mathcal{F}| \leq \binom{n}{\lfloor d/2 \rfloor}.$$

Az [5]-ös cikkben már be lett bizonyítva, abban az esetben, ha $n > 6(d+1)^2$.

Lineáris Programozási modellként fogom ezt a problémát is megközelíteni. Ahogy eddig is az x vektor elemeihez halmazokat fogok rendelni, és az alapján lesznek 0 és 1 értékűek, hogy bevesszük-e őket $\mathcal{F}$-be vagy nem. Ahhoz, hogy antilánc is legyen meg az átmérője legfeljebb d legyen $\mathcal{F}$-nek egy-egy feltételrendszert kell bevezetni.

Ahhoz, hogy antilánc legyen az $\mathcal{F}$ végeredményünk, úgy fogunk feltételt hozzáadni, hogy végigmegyünk az összes lehetséges A és B halmazpáron és ahol $A \subsetneq B$, ott:

$$x_A + x_B \leq 1.$$

Az átmérő, hogy legfeljebb d legyen úgy érhető el, hogy végigmegyek szintén minden lehetséges A és B halmazon és amelyiknél $|(A \backslash B) \cup (B \backslash A)| > d$ (a szimmetrikus differenciájuk legalább d):

$$x_A + x_B \leq 1$$

Tehát legfeljebb csak az egyik kerülhet be $\mathcal{F}$-be. A célfüggvény pedig a:

$$max\ \underline{1}x$$

Az összes lehetséges halmaz felett a $c = \underline{1}$ lesz, tehát csak a halmazok darabszámára lesz optimalizálva.

Azt lehet észrevenni, hogy mindkét esetben az összes, az alaphalmazból létrehozott részhalmazon, és azokból létrehozott halmazpáron végig kell menni, így a futási idő csökkentése érdekében a két feltételt egy cikluson belül adom hozzá.

A kódban a fentebb leírt feltételeket és egymásba ágyazott ciklusokat így lehet implementálni az $n = 8$ és $d = 7$ esetben:

```python
import pulp
# Initializing the base set, creating all subsets
base_Set = "A B C D E F G H".split()
max_Subset_Size = len(base_Set)
possible_Sets = [tuple(c) for c in pulp.allcombinations(base_Set, max_Subset_Size)]
setAll = set(possible_Sets)
x = pulp.LpVariable.dicts( name: "set", possible_Sets, lowBound=0, upBound=1, cat=pulp.LpInteger)
# Creating the model
set_Model = pulp.LpProblem( name: "Largest Antichain at fix maximal diameter", pulp.LpMaximize)
set_Model += pulp.lpSum([ x[it_set] for it_set in possible_Sets])
# Adding the constraints
for subSetA in setAll:
    for subSetB in setAll:
        subSetB = set(subSetB)
        subSetA = set(subSetA)
        if len(subSetA.symmetric_difference(subSetB)) > 7:
            subSetA = tuple(sorted(tuple(subSetA)))
            subSetB = tuple(sorted(tuple(subSetB)))
            set_Model += (pulp.lpSum([x[subSetA] + x[subSetB]]) <= 1 )
            subSetB = set(subSetB)
            subSetA = set(subSetA)

        if subSetA.issubset(subSetB) and subSetA != subSetB:
            subSetA = tuple(sorted(tuple(subSetA)))
            subSetB = tuple(sorted(tuple(subSetB)))
            set_Model += (pulp.lpSum([x[subSetA] + x[subSetB]]) <= 1 )
set_Model.solve()

print(f"The chosen sets are out of a total of {len(possible_Sets)}:")
for table in possible_Sets:
    if x[table].value() == 1.0:
        print(table)
```

6. ábra. Legnagyobb antilánc mérete adott maximális átmérő mellett

4.4. A legnagyobb antiláncot fix átmérő mellett kereső modell futása

A futási idő ebben az esetben körülbelül 210 másodpercig tartott és megtalálta az optimális értéket, ami összesen 56 darab halmazt talált, ami úgy állt össze, hogy az összes A-val kezdődő halmaz be lett véve, utána 5 elemű halmazok jönnek. Az összes halmazt be lehet venni, amiben nincs benne az A elem. Egy másik lehetőség 56 elemű halmazrendszer találására, ha az összes 5 elemű halmazt vesszük.

4.2. Megjegyzés. *A programot többször futtatva többféle konfigurációját megkaphatjuk ilyen halmazrendszereknek.*

Ez a halmazrendszer megfelel a sejtésnek, ugyanis:

$$|\mathcal{F}| = 35 + 21 = 56 \leq \binom{8}{3} = 56 \tag{4}$$

Ezen az érték kombináción kívül más értékeket is megnézek, $(8, 5)$ párra 155 másodpercig futott a megoldó, és végül 28 elemű halmazrendszert talált, ami

szintén megfelel a sejtésnek. A 6. fejezetben lefuttattam még értékpárokra, és táblázatba foglaltam az eredményeket.

Ezeket az értékeket és ennyi kombinációt végignézni kézzel nagyon komplikált, szinte lehetetlen lenne, de még minden halmazpárra ellenőrizni, átlátni, hogy tényleg teljesítik-e a feltételeket sem egyszerű, ezért külön hasznos ilyen modelleket használni.

5. Kleitman illesztési probléma

Ezzel a problémával a [2] cikkben találkoztam, és felkeltette a figyelmemet, hogy mennyi részeredmény született a témakörben, de a teljes megoldásra csak sejtések vannak. Az LP modellt is a cikkben találtam, amit a különböző tulajdonságú halmazrendszerek keresése közben sokszor módosítottam, és az általános halmazrendszer szerkezetével tett vizsgálatom is független az eredeti cikktől.

Legyen $s \geq 3$ egész szám, és jelölje $k(n, s)$ a maximum méretét a $\mathcal{F} \subset 2^{[n]}$, hogy ne legyen $\mathcal{F}$-ben s darab páronként diszjunkt halmaz.

Ez az érték már többféle esetre meg van határozva. Kleitman [6] cikkben meghatározta a $k(n, s)$ értéket akkor, ha $n \equiv 0$, vagy $-1 \pmod{s}$ a következő tételben.

5.1. Tétel (Kletiman [6]). *Legyen $s \geq 2$ egy pozitív egész szám, és $\mathcal{F} \subset 2^{[n]}$ egy halmazcsalád s darab páronként diszjunkt tag nélkül. Akkor minden $n = s(m+1) - l$, ahol $l \in [s]$, azt kapjuk, hogy:*

$$|\mathcal{F}| \leq \frac{l-1}{s}\binom{n}{m} + \sum_{t \geq m+1} \binom{n}{t}, \tag{5}$$

és ez a becslés éles $l \in 1, s$ esetén, tehát egyenlőséggel teljesül.

Emellett az is meg lett még mutatva, hogy abban az esetben ha s osztja n-et akkor: $k(n, s) = 2k(n-1, s)$.

Egy későbbi cikkben [7] Quinn meghatározta abban az esetben $k(n, s)$ értékét, amikor $n \equiv -2 \pmod{s}$ és $s = 3$. Ez ki lett terjesztve minden s értékre Frankl és Kupavskii által. Ekkorra az $s = 3$ teljesen le lett fedve, és $s = 4$ esetnél már csak egy maradékosztályra nem volt meghatározva az érték.

Ebből kiindulva Frankl és Tokushige megfogalmazta az alábbi sejtést [5] a fennmaradó 1 maradékosztályra.

5.1. Sejtés (Frankl, Tokushige [5]). *Legyen $s \geq 4$. Ha $n \equiv 1 \pmod{s}$, akkor*

$$k(n, s) = 4k(n-2, s). \tag{6}$$

A tételből adott, hogy $k(7, 4) = 120$, tehát 120 halmazt lehet kiválasztani, hogy ne legyen 4 darab páronként diszjunkt halmaz bennük 7 elemű alaphalmazon. A számok megválasztása és a módszer követése a [2] cikk alapján történt, de olyan eseteket is megemlítek, amiket a cikk nem foglal magába. Az lenne a cél ekkor, hogy felülről lehessen becsülni a $k(9, 4)$ értéket.

Ehhez a korábbiakhoz hasonlóan egy LP modellt fogok csinálni, amiben az x vektor elemeihez az összes $A \in 2^{[n]}$ halmazt fogom rendelni, aminek 0 lesz az értéke ha $A \notin \mathcal{F}$ és 1 ha $A \in \mathcal{F}$, tehát indikátora lesz. A feltételrendszer ezt a 4-es páronkénti diszjunkt tulajdonságot fogja vizsgálni. Minden A, B, C, D páronként diszjunkt halmazra az lesz a feltétel, hogy:

$$x_A + x_B + x_C + x_D \leq 3. \tag{7}$$

A cél pedig egyszerűen annyi lesz, hogy az x vektorban maximális legyen a változók összege:

$$\max \underline{1}x. \tag{8}$$

Azonban a 4 egymásba ágyazott végigiterálás a $2^{[9]}$ elemein, tehát mind az 511 (üres halmaz nem generálódik le, de az nem lenne bevéve) halmazon rengeteg lenne, akár napokon keresztül is futna a modell létrehozása, illetve a megoldó futása is hasonlóan megugorhat.

5.1. Heurisztika bevezetése a Kleitman féle illesztési problémára

Annak érdekében, hogy az előző $k(9, 4)$ értékének felülbecslésére vonatkozó problémát meg lehessen oldani be kell vezetni heurisztikákat. Ezekkel az lesz a cél, hogy visszább lehessen szorítani a modell létrehozására szükséges, illetve a megoldó által felhasznált időt. Azt fogjuk felhasználni, hogy

$$\forall A \subset [n], |A| \geq 4 : x_A = 1$$
$$\forall A \subset [n], |A| \leq 1 : x_A = 0.$$

A modell kódbeli implementálása után fogom megnézni, hogy ezek a feltételezések nem csökkentik-e a megtalált halmazrendszer méretét.

A fentebb leírt heurisztika magában nem lenne elég a futási idő csökkentésére, a 4 egymásba ágyazott ciklusban futási ideje hatalmas lenne, ezért ahogy a kódban is látni lehet, be kell azt is vezetni, hogy a 4 halmazra vonatkozó feltételek csak a heurisztikus felső becslésnél, 4-nél kisebb méretű halmazokra vezetem be a feltételeket, amik az alsó becslésnél, 1-nél nagyobbak. Fentebbi feltételekkel az alsó határnál kisebb halmazokat kizárom, azokra az x-beli érték mindig 0 lesz, amik a felső határnál nagyobbak, azokra meg 1 lesz, tehát mindig beveszem őket.

Kódban Kleitman illesztési problémája a heurisztikával kiegészítve így valósítható meg:

```python
import pulp
base_Set = "A B C D E F G H I".split()
max_Subset_Size = len(base_Set)
heurUP = 4
heurDO = 1
possible_Sets = [tuple(c) for c in pulp.allcombinations(base_Set, max_Subset_Size)]
setAll = set(possible_Sets)
x = pulp.LpVariable.dicts( name: "set", possible_Sets, lowBound=0, upBound=1, cat=pulp.LpInteger)

set_Model = pulp.LpProblem( name: "Kleitman Matching Problem Model", pulp.LpMaximize)
set_Model += pulp.lpSum([ x[it_set] for it_set in possible_Sets])

for subSet1 in setAll:
    if len(subSet1) >= heurUP: set_Model += ( pulp.lpSum(x[subSet1]) == 1 )
    elif len(subSet1) <= heurDO: set_Model += ( pulp.lpSum(x[subSet1]) == 0 )

for subSet1 in setAll:
    if (len(subSet1) > heurDO and len(subSet1) <= heurUP):
        for subSet2 in setAll:
            if (len(subSet2) > heurDO and len(subSet2) <= heurUP):
                for subSet3 in setAll:
                    if (len(subSet3) > heurDO and len(subSet3) <= heurUP):
                        for subSet4 in setAll:
                            if (len(subSet4) > heurDO and len(subSet4) <= heurUP):
                                subSet1 = set(subSet1)
                                subSet2 = set(subSet2)
                                subSet3 = set(subSet3)
                                subSet4 = set(subSet4)
                                if subSet1.isdisjoint(subSet2) and subSet1.isdisjoint(subSet3) and subSet1.isdisjoint(subSet4) and
                                    subSet1 = tuple(sorted(tuple(subSet1)))
                                    subSet2 = tuple(sorted(tuple(subSet2)))
                                    subSet3 = tuple(sorted(tuple(subSet3)))
                                    subSet4 = tuple(sorted(tuple(subSet4)))
                                    print('Added a constraint')
                                    set_Model += ( pulp.lpSum([x[subSet1] + x[subSet2] + x[subSet3] + x[subSet4]]) <= 3 )

set_Model.solve()

print(f"The chosen sets are out of a total of {len(possible_Sets)}:")
for table in possible_Sets:...
```

7. ábra. Kleitman illesztési problémája heurisztikával

Azt így nem lehet elkerülni, hogy rá kelljen futni olyan halmazokra, amiknek a mérete ki lett zárva, de a futás során olyankor nem megy tovább a beágyazott futásba, amivel rengeteg időt lehet így nyerni. A kódban szerepel két változó, $heurUP$ és $heurDO$ amik azt az értéket jelölik, hogy milyen méretű halmazokon nézzük, hogy ne lehessen s darab egymástól diszjunkt halmaz, előbbi a méret felső határát, utóbbi az alsó határát. A programban benne van még az elején a halmazok és a modell létrehozása, az x vektor, aminek elemei a halmazok és a korábbi fejezetekben szereplő modellekhez hasonlóan 0 és 1 közül veszik fel az egyik értéket és maximalizálni akarjuk az x-ben levő számok összegeit.

Utána a lehetséges halmazokon megyek végig, hogy megnézzem nagyobbak vagy egyenlőek-e a $heurUP$ értékkel, ami ebben az esetben 4, mert akkor az A halmazhoz $x_A = 1$. Amikor $|A| \leq heurDO$, tehát a legfeljebb 1 méretű

halmazoknál: $x_A = 0$.

Ezt követi a 4 egymásba ágyazott ciklus, amik egyenként végigmennek a halmaz összes elemén, de csak akkor lépnek eggyel bentebbi ciklusba, ha $heurDO \leq |A| \leq heurUP$. Legbelülre akkor ér be a program, ha mind az $s = 4$ halmaz mérete megfelelő lesz, akkor megnézem, hogy az összes halmaz páronként diszjunkt-e és ha ez teljesül, akkor hozzáadjuk az $x_A + x_B + x_C + x_D \leq 3$ feltételt, hogy nem választhatjuk be ilyen halmaznégyesek közül mind a négyet.

Ezt követően pedig már csak a modell megoldása és eredményének kiírása szerepel a kódban.

5.2. Kleitman illesztési probléma modellének futtatása

A program futási ideje több órás, de ebből mindössze $32 - 33$ másodperc az az idő, amíg az LP probléma megoldása történik. A maradék idő a négy egymásba ágyazott ciklusban telik el, még úgyis, hogy a halmazok, amikre a feltételek a méretükből adódóan alapvetően érvényesülnek vagy alapvetően nem érvényesülhetnek, több mint fele a mérettartományon kívül esik: A halmazok ahol $|A| \leq heurDO \vee |A| \geq heurUP$. Ennek ellenére a futási időnek a jelentős része itt telt el.

Eredménye pedig az lett, hogy sikerült 481 halmazt találnia. Ennyivel még nem lehetne teljesen biztos, hogy a feltételeknek megfelel ez a halmazrendszer, ugyanis az előfordulhatna, hogy mivel kevesebb a feltétel, nincs minden halmaz négyesre felvéve, hogy a heurisztikus felső becslésnél nagyobb méretű halmaz a végeredményként kapott $\mathcal{F}$ halmazrendszerben a szintén beválasztott kisebbekkel együtt véve mind a négy diszjunkt lenne, de ezt egy egyszerű számítással megcáfolhatjuk. A nagy halmaz mérete legalább 4-nek kell hogy legyen, mert a három méretű halmazokra fel van véve a modellbe a feltétel. A kis halmazok mérete amik $\mathcal{F}$-be kerülhetnek pedig legalább 2, így ha vennénk egy rendszert ezekből akkor az összesen legalább $3 \cdot 2 + 4 = 10$ elemet tartalmazna, tehát lenne olyan elem a skatulyaelv miatt ami két halmazban is benne lenne.

A kapott halmazrendszer így áll össze:

$$\mathcal{F} = \binom{[9]}{\geq 3} \cup \{A \in \binom{[9]}{2} : |A \cap [2]| \geq 1\}.$$

Ez összesen $466+15 = 481$ halmaz amit úgy kaptam, hogy minden legalább 3 méretű halmazt beleveszünk és a kettő méretű halmazok közül azokat, amik az első két elem közül legalább az egyiket tartalmazzák. Ez azért lesz a feltételeknek megfelelő halmazrendszer, mert két esetet különböztetek meg, hogy kettőnél több, vagy legfeljebb kettő 2 méretű halmaz szerepel tetszőlegesen

kiválasztott 4 halmaz között. Az első esetben, amikor legalább három 2 méretű halmaz van, akkor azért lesz biztosan metsző halmaz, mivel a két elemű halmazoknak legalább az egyik elemük a [2]-ben szerepel, tehát a három darab kételemű halmaz közül legalább kettő metszi egymást. A másik esetben pedig amikor legfeljebb két darab 2 méretű halmaz van, akkor összesen legalább $2+2+3+3=10$ elem van bennük összesen, ami azt jelenti a 9 elemű alaphalmaz miatt, hogy van olyan elem ami két halmazban is szerepel.

5.3. A halmazrendszer méretének felső becslése

Sikerült találni egy ellenpéldát a kiindulási sejtésre, de felmerülhet a kérdés, hogy $k(9,4)=481$, nem lehet-e ennél a 481 elemű halmaznál nagyobbat találni. Mivel minden legalább 2 méretű halmazt végignézett a program, így az merülhet fel, hogy 1 méretű halmaz is bekerülhetne az $\mathcal{F}$ halmazrendszerbe (ezt az esetet a $heurDO$ változót 0-ra véve meg lehetne oldani, de elemibben fogom megnézni a problémát), de akkor, ha azt az elemet leválasztjuk, ezt lehetne felírni:

$$|\mathcal{F}| \leq 2^{9-1} + k(9-1,3) \tag{9}$$

Úgy becsültem meg felülről a halmaz méretét, hogy vettem azt az elemet tartalmazó összes halmazt, és hozzáadtam e nélkül az elem nélkül és eggyel kisebb s-sel a legnagyobb $\mathcal{F}$ méretét, ami olyan, hogy bármely $s-1$ elem között található legalább két halmaz, amik metszőek, tehát a $k(8,3)$ értékét. Ennek az értéke már meg lett határozva az 5.1 tételben, ugyanis ott pontos a becslés, de a programom kódját is lehet úgy módosítani, hogy megtalálja ezt az értéket. Most megmutatom, hogy a programom és ezekkel a módszerekkel, hogyan lehet egyszerűen meghatározni ezt az értéket, a pontos kódbeli változtatások a [1]-ben szereplő linken megtekinthetőek. Eltávolítottam egy ciklust és a feltételeket átírtam, hogy minden páronként diszjunkt A,B,C halmazra:

$$x_A + x_B + x_C \leq 2$$

A futási ideje a programnak teljes egészében nagyjából 5 másodperc volt. Ez is azt szemlélti mekkora különbség van a különböző méretű modellek futási idejei között. Végeredményként a modell összesen 219 halmazt talált meg, az összes olyat belevéve, amelynél $|A| \geq 3$.

Így azt kaptam meg, hogy ha legalább egy 1 méretű halmaz is benne lenne $\mathcal{F}$-ben akkor legfeljebb $256+219=475$ halmaz lehetne $\mathcal{F}$-ben. Illetve azt is kipróbáltam, hogy ha egy $|A|=3$ halmazt megtiltok úgy, hogy a hozzá tartozó változót kizárom, hozzáadom az $x_A=0$ feltételt, akkor mindössze

480 méretű halmazrendszert talál, tehát mindössze azt az egy halmazt zárja ki, nem egy másik kételemű halmazokból álló rendszer lesz az eredmény.

A felső becsléseknek egy másik megközelítését fogom alkalmazni, bevezetek egy újabb definíciót, megkötöm a halmazok méretét.

5.1. Definíció. *A korábbi $k(n,s)$ fogalmat kibővítem úgy, hogy $k_i(n,s)$ jelölje a maximum méretét a $\mathcal{F} \subset 2^{[n]}$, és $\forall A \in \mathcal{F} : |A| = i$, hogy ne legyen $\mathcal{F}$-ben s darab páronként diszjunkt halmaz.*

Ehhez az új $k_i(n,s)$ fogalomhoz szintén átalakítottam [1] az eredeti LP-t megoldó programot. Korábban a 4.1-es fejezetben használtam szintén megméretezett alaphalmazt, ugyanazt a metódust fogom itt is, ezzel egyetemben a halmazrendszert (Pythonban „tuple" típus), aminek az elemeire optimalizál a program, az eredetiben szereplő összes halmazra optimalizáló beállításáról kicserélem, illetve a heurisztikát figyelmen kívül hagyom.

Az új méretezett halmazokra levő LP problémát mindössze a $k_2(9,4)$ esetre érdemes lefuttatni, mert a többi esetre már tudjuk, hogy lehet a teljes rendszert venni, az 1 méretű halmazokból pedig 3-at lehet venni, mert bárhogy legyen kiválasztva egy újabb halmaz az diszjunkt a többihez képest. A $k_2(9,4)$-re lefuttattam a módosított programot, a futási idő nagyjából 15 másodperc, aminek a nagy része az LP megoldásával telt és az optimális halmazrendszer mérete pedig $k_2(9,4) = 21$ amit sikerült így megtalálni. A halmazrendszer pedig szinte ellenkezőképp áll össze, mint ahogyan a $k(9,4)$ esetén volt a 2 méretű halmazok rendszere:

$$\mathcal{F} = \{A \in \binom{[9]}{2} : |A \cap [2]| = 0\}.$$

Erről a halmazrendszerről, hogy tényleg megfelel a feltételeknek hasonlóképp lehet látni, mint a korábbi esetben, a skatulyaelvvel, mert ha veszünk 4 darab halmazt $\mathcal{F}$-ből akkor a 7 elemű alaphalmazból az egyik elemet két különböző halmazban is ki kellett választani.

Ekkor felírható, hogy $k(n,s) \leq \sum_{i=1}^{n} k_i(n,s)$. Ezzel független szintekre lett osztva a $2^{[n]}$, a Boole-háló (3. ábrán 4 elemű alaphalmazra szerepel), halmaz méret alapján. Ekkor mindegyik szinten megkeressük a lehető legnagyobb ilyen rendszert, amik végül össze lesznek adva. A korábbi $k(9,4)$ értéknél így összesen 9-cel nagyobb értéket kapunk, ugyanis a 2 méretű halmazok közül 6-tal, az 1 méretűek közül 3-mal többet számolunk bele, mint a $k(9,4)$ esetében.

5.4. Az optimális halmazrendszer szerkezete

Abból kiindulva, hogy a Boole-hálóban a k. szinten $\frac{n!}{k!\cdot(n-k)!}$ darab halmaz van, azt lehet látni, hogy ahogy a középső szint fele van közelítve, nagyon gyorsan fog nőni. A $k(9,4)$ megtalálása során azt lehetett látni, hogy a legalább 3 méretű halmazokat mind be lehet venni, és a 2 méretűekből kellett kevesebbet, mint ami maximálisan lehetséges lenne, amelyet az LP optimális megoldása talált meg. Ebből azt feltételezem, hogy van egy szint, ami nem lesz teljes, és minden felette levő szintről viszont az összes halmaz be lesz választva.

Az 5.1 tételben [6] is hasonló van állítva, hogy van egy szint, amitől kezdve minden egyes halmaz benne van $\mathcal{F}$-ben. A tételben a kijelölt szint az az m. szint volt, ez van felülről becsülve $\frac{l-1}{s}\cdot\binom{n}{m}$-mel, ahol $l = s(m+1) - n$ tehát a maradék ha s méretű részekre osztanánk n-et felülről közelítve. Akkor a felső becslésben az összes m méretű halmaz számát megszorozzuk azzal, hogy hányszor férhetne bele a maradékba az $l-1$ az s-be arányaiban. Ami azt jelzi, hogy legfeljebb milyen arányban lehetnek m méretű halmazok tetszőlegesen kiválasztott s halmaz közül, hogy biztosan legyen közös elemük a halmazoknak. Az m-ik szint pedig a $\lfloor\frac{n}{s}\rfloor$ lesz ugyanis úgy gondolom, hogy a fentebbi szinteken sokkal több halmazt lehet bevenni az $\mathcal{F}$ halmazrendszerbe, de akkor az m. szinten már nem lehet mindent.

A $k(9,4)$ esetre ez azt a felső becslést adja, hogy az $m = 2$ méretű halmazok legfeljebb $\frac{3-1}{4}\cdot\binom{9}{2} = \frac{1}{2}\cdot 36 = 18$-an lehetnek, ami jobb becslés mint a $k_2(9,4) = 21$, a végeredményre ami 15.

5.1. Megjegyzés. *Erdős-Ko-Rado tétel ennek az egész problémakörnek az $s = 2$ esetét fedi le, ugyanis akkor bármelyik két halmazt $\mathcal{F}$-ből kiválasztva nem lehetnek diszjunktak, ami így egy legnagyobb metsző halmazrendszert fog adni. Ebben az esetben a* 4.1 *tételben ki van mondva, hogy a csillagszerű halmazrendszer tartalmazza a legtöbb halmazt.*

5.2. Megjegyzés. *Végignézem maradékosztály szerint rendezve, kik által lettek meghatározva a $k(n,s)$ értékek. Az $s = 3$ esetnél az n értékét* $\bmod 3$ *szerint osztom maradékosztályokra* 0 *és* 2 *esetén a [6] cikkben szereplő tétel ad pontos értéket, ezt Quinn [7] bővítette ki az $n \equiv -2 \pmod{s}$ esettel, így tehát már az összes eset le lett fedve amikor $s = 3$.*

Amikor $s = 4$ akkor hasonlóan a 3*-as esethez az* 5.1 *tétel megadja az optimális halmaz méretét a* 0 *és* 3 *maradékosztályokra. Az $n \equiv -2 \pmod{s}$ esetben minden s-re (Quinn eredménye után $s = 3$-ra) meghatározta Frankl és Kupasvskii a $k(n,s)$ értéket. A megmaradt* 1*-es maradékosztályra fogalmazódott meg az a* 5.1*-es sejtés [5], amire ellenpéldát Wágner Ádám talált [2].*

Még egy sejtést le szeretnék írni, amit Kupavskii közvetlenül javasolt a témavezetőmnek, és javaslatára néztem meg alaposabban a [10] cikkből, aminek ilyen halmazcsalád méretét és szerkezetét adja meg első sejésként, megadott paraméter értékek mellett.

5.2. Sejtés (Frankl, Kupavskii [10])**.** *Legyen $s \geq 2$, $m \geq 1$ és $l = (m+1) \cdot s - n$. Ha $0 < l \leq \lceil \frac{s}{2} \rceil$, akkor*

$$k(n,s) = \{P \subset 2^{[n]} : |P| + |P \cap [l-1]| \geq m+1\}$$

teljesülni fog.

5.3. Megjegyzés. *Érdemes megjegyezni, hogy a sejtés nem fedi le az összes lehetséges (n,s) értéket, illetve ennél egy erősebb állítást fogok megfogalmazni, hogy tudok olyan értéket mutatni amikor $l = 3$, ami kívül esik a sejtés feltételein és nem teljesíti a feltételeket $k(9,3)$ esetén.*

Példaképp megnézem a $k(9,4)$ esetet, ahol végignézem mit kapok az adott értékekre és, hogy hogyan épül fel a halmazrendszer. Kiszámolom a sejtésben szereplő változók értékeit: $n = 9$, $s = 4$, $m = \lfloor \frac{n}{s} \rfloor = 2$, és $l = 3$. A halmazrendszer, amit a fentebbi sejtés alapján kapni lehet úgy fog felépülni, hogy azok a halmazok fognak bekerülni, amik legalább 3 méretűek vagy az adott halmaz méretét az első két elemmel vett metszet méretéhez véve legalább 3 lesz.

Ez pontosan ugyanazt fogja adni, mint amit fentebb leírtam a megfigyelt halmazrendszerről, amikor a $k(9,4)$-re futtattam az LP modellt.

Készítettem egy külön programot, ami megadja a fentebbi sejtésben szereplő halmazrendszert:

A kódot nem fejtem ki részleteiben, a fentebb leírtakat követi, és annak a leprogramozott változata, a futása sokkal gyorsabb az LP megoldóknál tapasztalt futási időknél, néhány másodperc alatt megtalálja a halmazrendszer méretét 21 méretű alaphalmaz esetén, amíg ilyen méretű alaphalmaz esetén évekbe telne lefutnia az LP megoldónak.

Ahhoz, hogy össze tudjam hasonlítani a két külön módszert, lefuttattam minden esetet $k(11,4)$-nél, és $k(11,3)$-nál kisebb esetre, ehhez kellett további heurisztikát tenni, hogy lefussanak ekkora modellek, így figyelmen kívül hagytam az $m = 4$. szinttől kezdve mindent, ami a modellben szerepelt. A változtatások ehhez nem igényeltek nagy kódbeli átírásokat, mindössze a $possible_sets$ nevű változó létrehozásakor megadtam, hogy legfeljebb 3 méretű halmazokkal foglalkozzak. Miután el tudtam érni, hogy nagyobb modellek is lefussanak az eddigieken felül, megnéztem a talált halmazrendszereket, és pontosan egyeztek az értékek és halmazok a sejtésben meghatározottakkal.

```python
import pulp
import numpy as np
base_Set = "A B C D E F G H I".split()
max_Subset_Size = len(base_Set)
s = 4
m = np.floor(max_Subset_Size / s)
l = (m+1) * s - max_Subset_Size
l = int(l)
print(type(l))
possible_Sets = [tuple(c) for c in pulp.allcombinations(base_Set, max_Subset_Size)]
lSet = []

for i in range(l-1):
    lSet.append(base_Set[i])

conjSets = []

for setForSize in possible_Sets:
    lSet = set(lSet)
    setForSize = set(setForSize)
    if (len(setForSize) + len(setForSize.intersection(lSet))) >= m+1:
        setForSize = tuple(sorted(tuple(setForSize)))
        conjSets.append(setForSize)
    lSet = tuple(sorted(tuple(lSet)))
    setForSize = tuple(sorted(tuple(setForSize)))

print('Total number of sets: ',len(conjSets))
for set in conjSets:
    print(set)
```

8. ábra. Kleitman illesztési problémája heurisztikával

Egyedül a $k(9,3)$ esetben, amikor $l = 0$ így nem tartozik bele a sejtésbe, találtam különbséget. Az LP által megtalált optimális halmazrendszer 438 méretű, amit ellenőriztem a nagyobb szintekkel bevett feltételekkel. A sejtés által talált halmazrendszer viszont csak 432 méretű.

A $k(9,3)$ esetet megvizsgálom alaposabban, hogy hol tér el az 5.2 sejtésben mondott szerkezetétől a halmazrendszereknek. A megfelelő paraméter értékek a következők: $n = 9$, $s = 3$, $m = 3$, $l = (3+1)3 - 9 = 3$.

Ebben az esetben a fentebbi módszer azt adja, hogy a halmaz és az első 2 elemmel vett metszet mérete legalább 4 kell, hogy legyen, de a gyakorlatban lefuttatva a konkrét LP modellt azt kaptam, hogy elég mindössze egy elemet kihagyni, és akkor az összes 3 elemű halmazt kell venni, ami 6-tal több halmazt fog eredményezni, mintha a sejtésbeli konstrukció lett volna követve, mert $\binom{8}{3} = 56$, míg a másik módszerrel $\binom{7}{2} \cdot 2 + 7 + 1 = 50$ féle halmaz létezik.

A $k(9,3)$ feltételeit teljesíti a kiterjesztettebb halmazrendszer is, ugyanis akkor nem teljesíthetné, ha mindhárom halmaz, amit kiválasztunk 3 méretű, de akkor mivel azok 8 elemű alaphalmazra lettek szűkítve, így a skatulyaelv miatt nem lehetnek mind diszjunktak.

A sejtésben az l értékére szerepelt felső becslés is, erre nem találtam olyan esetet amikor nem teljesülne, csak arra amikor az alsó becslés sérül.

6. Korábbi LP modellek futásának gyorsítása

A szakdolgozatomban többször is kitértem a futási időre, ami nagyon nem elhanyagolható tényező halmazrendszerek keresésekor, ugyanis könnyen exponenciálisan tud növekedni egy túl nagy modell esetében. A számítógépemen a futási idők nagyban tudnak változni, kifejezetten párhuzamos futtatáskor, de a megadott időknél próbáltam a legkisebb időt megadni, amit el tudtam érni. Egy másik fontos tényező a futási idő tekintetében maga az LP megoldásához alkalmazott heurisztika, például a Kleitman-féle illesztési probléma modelljében használtam korábban heurisztikát, ugyanis anélkül nem futott volna le napokon keresztül a program, és hiába lett sokat egyszerűsítve a modell, úgyis meg lehetett mutatni, hogy az az optimális méretű. A futási idők közti különbséget táblázatban fogom összefoglalni a különböző részfejezeteknél. A kódokat ebben az esetben is átalakítom, mindegyiknél az elején és a megoldó futása után kiiratom az aktuális időt, hogy pontosan meg lehessen mondani a közben eltelt időt, ehhez most is a datetime nevű Python könyvtárat fogom használni.

6.1. Kleitman illesztési probléma futásának gyorsítása

Sorrendben visszafelé fogok haladni a szakdolgozatban, így a legutóbbi, 5. fejezetetbeli LP modell futásával fogok foglalkozni.

Amit az eredeti [2] cikkhez képest másképp fogok csinálni, hogy az egymásba ágyazott ciklusban nem minden halmazon fogok végigmenni, hanem csak azokon, amik legalább $heurDO$, és legfeljebb $heurUP$ méretűek, ez jelen futtatásoknál 2 és 4. Ettől jelentős gyorsulást várok, ugyanis eddig az idő legnagyobb része az egymásba ágyazott iterációkban telt el.

A kód módosítása utáni futtatás mindössze körülbelül négy és fél percig tartott, míg anélkül nagyjából hét és fél percig. Másik a fejezetben felhasznált LP problémát is megnézek, hogy milyen gyorsan fut le.

1. táblázat. Kleitman illesztési probléma futási idejei

Eredeti $k(9,4)$ probléma heurisztikával	450 másodperc
$k(9,4)$ **probléma,** $2,3$ **méretű halmazokon iterálva**	270 másodperc
$k(8,3)$ **probléma**	4 másodperc
$k(9,3)$**probléma**	1093 másodperc

Meglepő volt, hogy a heurisztikával ellátott $k(9,3)$ esetnél a pulp könyvtár megoldója jelentősen lassabban futott le a $k(9,4)$ esetnél ahol a legtöbb idő a

modell létrehozásához szükséges iterációkkal telik és mindössze 30 másodperc az az idő amíg a megoldó fut.

6.2. Legnagyobb antiláncot fix átmérő mellett kereső modell futási ideje

A 4. fejezetben csak az $(n, d) = (8, 7)$ esetre futtattam le a modellt. A modellben a feltételek hozzáadásában két ciklus van egymásba ágyazva, amik ilyen méretű halmazok mellett a teljes futási időhöz képest hamar lefutnak, az LP probléma megoldásával telt a legtöbb idő, és ez fog a táblázatban szereplni.

2. táblázat. A modellek megoldási idejei az (n, d) értékek függvényében

$(8, 5)$ **esetén a futási idő**	155 másodperc
$(8, 3)$ **probléma, warmstart nélkül**	3800 másodperc
$(8, 3)$ **probléma warmstarttal**	3202 másodperc
$(9, 8)$ **probléma warmstarttal**	89 másodperc (112 warmstart nélkül)
$(10, 3)$ **probléma warmstarttal**	$>$ 36 óra (nem futott le)
$(9, 7)$ **probléma warmstarttal**	$>$ 36 óra (nem futott le)

A [2] cikkben szereplő $(n, d) = (10, 3)$ esetet próbáltam reprodukálni, először a leírt módon, de nem futott le 24 órán belül a számítógépemen. Nagy különbség tud kialakulni ennél a problémánál az LP megoldó által szükséges időkben, a táblázat alapján is látszik, hogy a nagyobb maximális diverzitás segíti a gyorsabb megoldást. Azt is kipróbáltam, hogy a modellbeli x vektornak beadtam kezdeti értékeket, ez van „warmstart"-nak hívva. Alapesetben a kezdeti értékeket mind 0-ra állítottam, kivéve az egy méretű halmazokat, mert azokra 1-et adtam be, és azt figyeltem meg, hogy nagyjából $20-25\%$-os gyorsulást tudtam elérni. A $(8, 3)$ esetben 8 méretű, $(9, 8)$ esetben pedig 126 méretű az optimális halmazrendszer, amit a modell talált, ezeket ellenőriztem és megfelelnek a sejtésben tett feltételezésnek.

6.3. Uniformizált halmazrendszer diverzitásának becslése

Ebben az esetben egy esetre futtattam le a modellt, az $(n, k) = (7, 3)$-ra, és a teljes program nagyjából 2 másodperc alatt futott le, ebből kevesebb mint 1 másodperc volt az LP megoldása.

6.4. Sperner rendszer találásának futási ideje

3. táblázat. Sperner rendszerek megtalálásának ideje

9 **méretű alaphalmazon**	33 másodperc
10 **méretű alaphalmazon**	125 másodperc
11 **méretű alaphalmazon**	490 másodperc

Ezeken a példákon, ahogy korábbiaknál is meg lehetett figyelni általánosan, hogy az alaphalmaz méretének növelésével exponenciális lesz a futási idő növekedése, ami két tényezőből áll össze. Az egyik az, hogy a modell létrehozásánál végig kell menni a halmazokon egymásba ágyazottan, hogy minden halmazpárra legyen feltétel, illetve utána pedig a lineáris programozási feladat megoldásának is nőtt az ideje, de azt figyeltem meg, hogy nagyobb ütemben, mint a ciklusok mérete.

Összefoglalás

A szakdolgozatom során megnéztem, hogyan lehet különböző kombinatorikai problémákat modellezni és számítógéppel megoldani. Az elején a gráfok témakörében történő szélsőérték feladattal vezettem fel a kombinatorikai problémákat, amit a későbbi fejezetek során általánosítottam és bevezettem a halmazrendszereket, majd egyre komplexebb struktúrákat, amikre az adott feltételek mellett optimálizálni kellett. Ezek során megállapítottam halmazrendszerek maximális méretét a leírt modelleket felhasználva, és ezáltal több sejtést is sikerült megcáfolni, vagy másik esetben, ha nem sikerült cáfolni, akkor az azt jelenti kis értékekre biztosan teljesül az adott sejtés. Ezekhez Wágner Ádám Zsolt [2] cikkjét használtam fel amiben ezek szerepeltek, de tettem külön megfigyeléseket is ezek kapcsán. Azért lehetett csak kisebb értékekre futtatni a modelleket, mert a Lineáris Problémákban az összes lehetséges $A \in 2^{[n]}$ halmaz be van véve az x vektorba és a megoldó program megnézi bevegye-e, de az exponenciálisan nő, így hamar olyan méretű feltételrendszer tud kialakulni, ami már túl nagy a megoldónak, ezt le lehet csökkenteni az adott probléma függvényében, de a lehetőségek száma úgy is nagy ütemben nő.

Ezáltal minden lehetséges halmazt át lehet nézni a számítógéppel és a végén megtalál olyan halmazrendszereket, amiket kézzel szinte lehetetlen lenne végignézni, megtalálni, emiatt úgy gondolom hasznos eszköze lehet különböző kombinatorikai problémák megoldásának számítógép segítségének igénybe vétele.

Hivatkozások

[1] Githubon a modellek kódjához link

[2] Adam Zsolt Wagner, REFUTING CONJECTURES IN EXTREMAL COMBINATORICS VIA LINEAR PROGRAMMING, https://arxiv.org/pdf/1903.05495.pdf, 2019

[3] P. Erdős, Chao Ko, R. Rado, Intersection theorems for systems of finite sets, Quart. J. Math. Oxford (2) 12 (1961) 313-320.

[4] Nathan Lemons and Cory Palmer, The unbalance of set systems, Graphs and Combinatorics 24 (2008), no. 4, 361–365.

[5] P. Frankl and N. Tokushige, Extremal problems for finite sets, Student mathematical library, American Mathematical Society, 2018.

[6] Daniel J Kleitman, Maximal number of subsets of a finite set no k of which are pairwise disjoint, Journal of Combinatorial Theory 5 (1968), no. 2, 157–163.

[7] F. Quinn, PhD thesis, (1986).

[8] P. Turán, On an extremal problem in graph theory, Mat. Fiz. Lapok (1941).

[9] P. Frankl, Antichains of fixed diameter, Moscow Journal of Combinatorics and Number Theory 7 (2017), N3.

[10] Peter Frankl, Andrey Kupavskii, Families with no s pairwise disjoint sets, https://arxiv.org/pdf/1607.06122.pdf