Kondor Gábor

Egyoldali párosítási piacok egyenletes klaszterezési megközelítésben

Pénzügy Tanszék

Témavezető:
Dr. Vidovics-Dancs Ágnes

Budapesti Corvinus Egyetem

Közgazdasági és Gazdaságinformatikai Doktori Iskola

Egyoldali párosítási piacok egyenletes klaszterezési megközelítésben

Doktori értekezés

Kondor Gábor

Budapest, 2022

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok és algoritmusok jegyzéke

Köszönetnyilvánítás

Nemo vir est qui mundum non reddat meliorem.

Mennyei Királyság

Ezúton szeretnék köszönetet mondani témavezetőmnek, Vidovics-Dancs Ágnesnek, aki szakmai iránymutatásaival és tanácsaival segítette a dolgozat elkészültét, és emelte annak színvonalát.

Köszönöm a támogatást a Befektetések és Vállalati Pénzügy Tanszék kollégáinak, és különösen Csóka Péternek a számos közös szakmai munkát.

Hálásan köszönöm a sok támogatást Editnek, Családomnak, Vali néninek és András bácsinak.

A dolgozatban maradt hibákért természetesen kizárólag én vagyok felelős.

A doktori képzésem során a Pallas Athéné Domus Educationis Alapítvány ösztöndíjas hallgatója voltam. A dolgozatban szereplő elemzést az MNB Kutatási Kiválósági Díj pénzügyi támogatásával vásárolt asztali számítógépen végeztem.

1. fejezet

Bevezetés

A közgazdaságtan egyik fontos kérdése, hogy az egyes piacok hogyan allokálják az erőforrásokat. A párosítási piacokon nincs, vagy csak részben van olyan árrendszer, amely meghatározza az allokációkat, és a létrejövő párosításokat elsősorban a piacot szabályozó mechanizmusok adják meg. (Nobel Prize, 2012b)

A párosítási piacok elméletének alapjait Gale és Shapley (1962) fektették le[1], akik a párosítási feladat megoldására a *stabilitás* koncepciót javasolták. Ennek megfelelően kétfős párok esetében olyan párosítás kialakítása a cél, amelyben nincs két olyan különböző párokban lévő szereplő, akik jobban preferálják egymást, mint saját párjukat. Az egyoldali párosítási piacok egyik alapvető modellje a Gale és Shapley (1962) által meghatározott *stabil szobatársak probléma*, amelyben a szereplőknek szigorú preferencia-rendezése van a többiekre nézve, és a feladat egy stabil párosítás létrehozása, amennyiben az létezik.

Az egyoldali párosítások egyik fontos alkalmazási területe a vesecsere-programok (lásd pl. Biró (2006)). Ezek esetén olyan beteg-donor párok a piaci szereplők, amelyek egyik tagjának új vesére van szüksége, a másik tagja pedig hajlandó lenne donorként odaadni az egyik veséjét, ugyanakkor egymással, a transzplantáció szempontjából, nem kompatibilisek. A feladat olyan párosítás kialakítása, amelyben az egymáshoz társított két különböző beteg-donor pár között keresztben valószínűsíthetően megvalósulhat a transzplantáció - későbbi tesztek során ennek még sajnálatos módon kiderülhet az ellenkezője.

Morrill (2010) az egyoldali párosítási feladat megoldásra a stabilitással szemben egy

[1] Az elmélet később számos gyakorlati alkalmazás alapjául szolgált, mint például orvosi rezidensek kórházakhoz rendelése, egyetemi felvételi eljárások, vesecsere-programok. A stabil allokációk elméletéért és a piactervezés területén végzett munkájukért Lloyd Shapley és Alvin E. Roth kapta a 2012-es közgazdasági Nobel-díjat (Nobel Prize, 2012a).

alternatív megközelítést, a Pareto-hatékonyságot javasolja. Érvelése szerint a szobatárs probléma esetében a stabilitás figyelmen kívül hagyja azt a meghatározó fizikai korlátot, hogy a szobatársaknak szobákra van szüksége, és ennélfogva az túlságosan szigorú. Ugyanis, hogyha egy szobabeosztás már kialakult, akkor hiába szeretne két különböző szobában lévő egyén összeköltözni, egyoldalúan egyikük sem költöztetheti ki a saját szobatársát, és az új párosítás nem jöhet létre. Ez különösen fontos lehet a vesecserék esetében, ahol a párosítás után is kiderülhet még inkompatibilitás.

Az eddig említett megközelítések párok kialakítsára vonatkoznak, ugyanakkor a gyakorlatban vannak olyan feladatok, amelyek esetében előfordulhatnak nagyobb méretű csoportok is. Például a vesecserék esetében van olyan program, amelyben megengedettek a hármas párok (Biró, 2006), a szobatársak tekintetében pedig gyakran 3- vagy 4-fős szobákkal találkozunk. A szobatárs problémának azt a változatát, amelyben háromfős szobák kialakítása a feladat, 3-dimenziós szobatárs problémának (3D-SR) nevezzük. Ismereteink szerint egy kivétellel (Arkin és szerzőtársai (2009) 2-stabil párosítás megközelítése) valamennyi 3-dimenziós felírás NP-teljes eredményre vezet. Ez azt jelenti, hogy ezen feladatok esetén bonyolultságelméleti szempontból nincs hatékony módszer a megoldás meghatározására, így a gyakorlatban nagyméretű problémák esetén nem tudjuk megadni azt. Továbbá megjegyezzük, hogy mindössze egy olyan megközelítéssel találkoztunk (Lam és Plaxton (2019) teljes ciklikus listákra vonatkozó eredménye), amely a háromnál magasabb dimenzió esetét is tárgyalja.

Segev és szerzőtársai (2005) a vesecserékre egy olyan gyakorlatban is alkalmazott megközelítést tekintettek, amely egy súlyozott párosítási problémára vezethető vissza (Biró, 2006). A szerzők szimulációs kísérletek alapján úgy találták, hogy módszerükkel országos szinten jobb eredményt lehetne elérni, mint az "első találatot elfogadó" párosítási eljárást alkalmazó vesecsere központok esetében.

A disszertáció célja röviden a következőképpen fogalmazható meg:

- A súlyozott párosítási megközelítés kiterjesztése egyoldali párosítási kontextusban a csoportok tetszőleges méretére. Kifejezett célunk olyan keret meghatározása, amelyben a minimalizálási és maximalizálási célokat egyaránt tudjuk vizsgálni.
- Az így megfogalmazott feladatok elméleti megoldhatóságának, vagy másképpen azok bonyolultságának vizsgálata.
- A feladatok gyakorlati megoldhatóságának és tulajdonságainak vizsgálata széleskörű

szimulációs kísérletek segítségével az irodalomban található heurisztikák és saját algoritmusok eredményeinek összehasonlításával.

A súlyozott párosítási feladat tetszőleges m csoportméretre történő kiterjesztésére az m-dimenziós párosítási problémát vesszük alapul. Ez egy gráfparticionálási problémaként kezeli a csoportosítási feladatot, amelyben Pareto-hatékony megoldást határozunk meg. A disszertácó középpontjában ennek egy speciális esete áll, amelyben a szereplők egy euklideszi tér pontjainak felelnek meg, a kapcsolataikat a közöttük adódó euklideszi távolságok adják meg, a cél pedig m főből álló csoportok létrehozása úgy, hogy a csoportokon belüli távolságok négyzetösszege a konkrét feladattól függően minimális vagy maximális legyen.

Vegyük észre, hogy a feladat során a célfüggvény értékéhez a csoporton belüli valamennyi élt figyelembe kell venni. Ebből következően a vesecserékre háromfősnél nagyobb csoportok kialakítására nem tudjuk interpretálni, hiszen ennél az alkalmazásnál tulajdonképpen körök kialakítása a feladat. Megjegyezzük továbbá, hogy a vesecserékre a gyakorlatban helyesebb lenne olyan megoldási koncepció, amely megengedi egyszerre a két- és háromfős csoportokat is, valamint figyelembe tudja venni a háromfős csoportok esetén a két különböző csere-irányt is.

Az előző megfontolások miatt az euklideszi térben felírt problémát szobatársak párosításaként, vagy másképpen kollégiumi szobákhoz rendeléseként kezeljük, és *m-szobatárs problémának* [2,3] nevezzük. Az euklideszi tér dimenziói a hallgatók tulajdonságait reprezentálják, és az euklideszi távolságok határozzák meg, hogy mennyire lennének jó szobatársak. A cél egy kollégium hallgatóinak beosztása egyenlő méretű szobákba úgy, hogy az számukra a legelőnyösebb legyen.

A csoportosítás módjára két különböző megközelítést tárgyalunk, amelyek mellett különböző érvek szólnak. Az egyikben azt tesszük fel, hogy olyan csoportok kialakítása a kedvező, amelyben hasonló egyének szerepelnek. Ekkor homogén csoportokról vagy klaszterekről beszélünk. Emögött a motivációt a hasonló érdeklődési területek és tulajdonságok révén feltételezhetően kialakuló jobb kapcsolatok adják. Ekkor egy minimalzálási problémát tekintünk. A másik megközelítésben ezzel ellenkezőleg, a napjainkban egyre nagyobb szerepet kapó diverzitás fontosságával összhangban, olyan szobabeosztásokat

[2]Vegyük észre, hogy az m-szobatárs probléma csupán elnevezésében hasonlít a Gale és Shapley (1962) által megfogalmazott szobatárs problémára. Az általunk tárgyalt feladat nem preferencia-sorrendekre épül, és nem cél stabil megoldás meghatározása.

[3]Ugyanerre a problémára Kondor (2018) konferenciaelőadásomban és 2015-ös k-szobatárs probléma című TDK dolgozatomban a k-szobatárs probléma elnevezést használtuk. Ugyanakkor, hogy konzisztensek legyünk a szélesebb szakirodalomban megjelenő, kapcsolódó problémákkal, ezt módosítottuk.

kívánunk megadni, amelyben a szobatársak a lehető legváltozatosabb tulajdonságokkal rendelkeznek. Ekkor heterogén csoportok kialakítása céljából egy maximalizálási problémát vizsgálunk.

Hagyományosan a *klasztereket* olyan csoportoknak tekintjük, amelyek elemei egymáshoz hasonlóak. Ebből eredően a mi esetünkben szigorúan véve az m-dimenziós párosítás és az m-szobatárs problémáknak csupán a minimalizálási verziója tekinthető egyenletes klaszterezési feladatnak, vagyis olyan problémának, amelyben a cél egymáshoz hasonló elemekből álló, egyenlő méretű csoportok kialakítása. A mi esetünkben a maximalizálási feladat célja diverz csoportok kialakítása. Ugyanakkor az irodalomban található olyan megközelítés (lásd például Feo és Khellaf (1990), valamint Feo és szerzőtársai (1992) által tekintett k-particionálás probléma), amelyben a szereplők közötti távolságokat kompatibilitási értékekként interpretálják, így a klaszterek kialakítását egy maximalizálási feladat írja le. Emiatt, és részben az egyszerűbb tárgyalás érdekében is, a *klaszterezés* elnevezést megengedően használjuk, és olyan csoportokként tekintünk a klaszterekre, amelyeket egy meghatározott minimalizálási vagy maximalizálási probléma megoldásaként alakítottunk ki. Ezzel összhangban pedig a dolgozat során az egyenlő méretű csoportok kialakításának problémáit összefoglalóan *egyenletes klaszterezési problémáknak* nevezzük.

Az azonos méretű csoportokat megkövetelő és általánosságban véve az elemszámkorlátokkal ellátott, kapcsolódó feladatok igen sokrétűek. A 2. Fejezetben részletes áttekintést adunk az alkalmazási lehetőségekről, amelyek magukba foglalják a különböző tanulói csoportok létrehozását (Weitz és Lakshminarayanan, 1996; Höppner és Klawonn, 2008), a változatos munkacsoportok kialakítását (Bhadury és szerzőtársai, 2000), valamint az erőforrások allokációjának legkülönbözőbb formáit, mint például a szoftverszolgáltatás (SaaS, Su és szerzőtársai (2015)), a vezetéknélküli szenzorhálózatok (WSN, Lan és szerzőtársai (2009); Liao és szerzőtársai (2013)) és a többügynökös utazóügynök probléma (MTSP, Nallusamy és szerzőtársai (2009)).

A 3. Fejezetben megalapozzuk az egyenletes klaszterezési problémák megoldhatóságának vizsgálatát. Ehhez először illusztráljuk az m-szobatárs probléma nehézségét az esetek számának megadásával. Ezt követően bevezetjük a csoportosítási problémák elméleti nehézségének vizsgálatához szükséges eszköztárat. A minimalizálási és maximalizálási feladatokat általánosan optimalizálási problémáknak nevezzük. Ennélfogva, főként Ausiello és szerzőtársai (2003) könyvére támaszkodva, megadjuk az optimalizálási problémákra

vonatkozó, a tárgyalás szempontjából releváns bonyolultságelméleti osztályokat és eredményeket.

Már a definíciók ismeretében áttekintjük a csoportosítási feladatok nehézségi eredményeit. Itt kitérünk mind az általános, mind az elemszámkorlátokkal rendelkező optimalizálási problémákra, köztük az m-szobatárs probléma minimalizálási és maximalizálási változataival rokon megfogalmazásokra is, úgy mint az m-dimenziós párosítás (Feo és Khellaf, 1990), a k-particionálás probléma (Feo és szerzőtársai, 1992), valamint az egyenletes MSSC probléma (Pyatkin és szerzőtársai, 2017), amely tulajdonképpen a népszerű k-közép klaszterezés egyenlő elemszámú csoportokra vonatkozó kikötéssel. A problémák elnevezésénél a k a kialakítandó partíciók vagy klaszterek számára, míg az m a csoportok egységes méretére utal. A mi megfogalmazásunk szerint ezek rendre a szobák, illetve a szobatársak száma.

A fejezet zárásaként Kondor (2022a) eredményeinek bemutatásával kiterjesztjük a kapcsolódó nehézségi eredményeket általános dimenzióban, és megmutatjuk, hogy az általunk tárgyalt párosítási problémák legalább háromfős, egyenlő méretű csoportok kialakítására NP-nehezek. Ez pedig a gyakorlatban azt jelenti, hogy nagyméretű problémák esetén jelenlegi ismereteink szerint ezekre a feladatokra nem tudjuk megadni az optimumot.

A 4. Fejezetben Kondor (2022b) alapján bemutatjuk a szobatársak beosztására Gale és Shapley (1962) szigorú preferenciarendezésekre építő felírását, a stabil szobatársak problémát, valamint tárgyaljuk ennek változatait és a kapcsolódó bonyolultságelméleti eredményeket. Ezután szintén Kondor (2022b) tanulmányára építve bevezetünk egy Pareto-hatékony megoldási koncepciót. Végül megadjuk az m-szobatárs probléma formális definícióját, valamint tárgyaljuk a modell előnyeit és hátrányait a stabil szobatársak probléma ellenében.

Az egyenletes klaszterezési problémákon belül számos olyan feladat található, amelyekre az optimum megadása általánosan nem lehetséges. Emiatt az optimális megoldás helyett a gyakorlatban egy olyan megengedett megoldást keresünk, amelyet ‘elég jónak’ gondolunk. A megengedett megoldás jelen esetben egy olyan klaszterezést jelent, amelyre teljesülnek a feladat által előírt korlátok, vagyis a meghatározott csoportok mérete azonos, ugyanakkor maga a megoldás nem feltétlenül optimális.

A megengedett megoldások konstrukciójának egyik lehetséges megközelítését, és a kapcsolódó eljárásokat az 5. Fejezetben mutatjuk be. Ezek olyan polinomiális futásidejű,

vagyis bonyolultságelméleti szempontból gyorsnak tekintett algoritmusok, amelyek valamilyen konkrétumot adnak a megoldás szuboptimalitására. Az irodalomban erre kétféle módszert találunk. Az elsőbe az approximációs eljárások tartoznak, amelyeknél ismert az algoritmus teljesítményi rátája, így tudjuk, hogy legfeljebb mekkora a megengedett megoldás és az optimum aránya. A másik módszer a kúp optimalizálás alkalmazása, amely konkrét alsó és felső korlátot ad az optimum értékére.

A feladat megoldásának a gyakorlatban gyakrabban alkalmazott, másik megközelítése, hogy olyan heurisztikus eljárást adunk meg, amelyről úgy gondoljuk, hogy eredményes lehet az optimalizálási cél elérésében. Ezek az algoritmusok általában gyorsan számíthatóak, ugyanakkor a futásidejüket nem tudjuk megbecsülni. A 6. Fejezetben bemutatjuk az irodalomban található legfontosabb algoritmusokat, és új módszereket is megkonstruálunk. Ezeknek szintén két csoportját tárgyaljuk.

Az elsőbe a klaszterelemzéshez kapcsolódó eljárások tartoznak, amelyek alapvetően alacsony futásidővel rendelkeznek, ugyanakkor nem garantált, hogy a megoldásban a csoportok száma egyenlő lesz. Hogy az m-szobatárs problémára ezek alkalmazhatóak legyenek, megfogalmazunk 6 különböző heurisztikus eljárást a klaszterek kiegyenlítésére, amelyeket a 7. Fejezetben tesztelünk.

A második csoportban olyan eljárásokat tárgyalunk, amelyek konstrukciójukból adódóan alkalmasak egyenlő elemszámú klaszterek előállítására. Ezek közül többnek is közös vonása, hogy párok cseréjével próbál meg javítani a megoldáson. A mi hozzájárulásunk az eljárások ezen csoportjához, hogy megvizsgáljuk a nagyobb méretű cserék lehetőségét, és megfogalmazunk három heurisztikus eljárást, amelyek kettes és hármas cserék segítségével keresik az optimumot. Ezeket szintén bevonjuk a 7. Fejezetben elvégzett elemzésbe.

A 7. Fejezet során végrehajtjuk a probléma egy több lépésből álló vizsgálatát egy széles elemzési keretben. Az irodalomban rendszerint az optimalizálási problémáknak csak az egyik, vagy a minimalizálási vagy a maximalizálási verzióját tekintik, és erre értékelik ki a heurisztikák egy szűkebb halmazát. A mi esetünkben egyszerre vizsgáljuk a feladat mindkét megfogalmazását, és az eljárások széles körét bevonjuk az elemzésbe.

Célunk egyrészt, hogy jobb betekintést nyerjünk a probléma természetébe. Ehhez egy általunk konstruált eljárással a kisebb hallgatói létszámú esetekben a lehetséges szobabeosztások össztávolságainak hisztogramját is megjelenítjük. Másrészt, szeretnénk meghatározni az általunk javasolt kiegyenlítő módszerek és hármas cserék hozzáadott értékét az

irodalomban szereplő eljárásokhoz. Hogy ezt meg tudjuk állapítani, nagyszámú mintát generálunk nagy hallgatói létszámmal rendelkező esetekből, és ezekre tekintjük a feladat minimalizálási és maximalizálási, valamint a kis- és nagy csoportszámokkal rendelkező variációit. Nagyméretű problémák esetében párok kialakítására is teszteljük az eljárásokat, ahol azok eredményeit az optimális megoldással is össze tudjuk hasonlítani.

A 8. Fejezetben összegezzük a dolgozat eredményeit, és további kutatási irányokat jelölünk meg.

2. fejezet

Alkalmazási lehetőségek

E fejezetben áttekintést adunk a széleskörű alkalmazási lehetőségekről az irodalomban megfogalmazott, kapcsolódó feladatokon keresztül. Ezzel egyrészt összekapcsoljuk az m-szobatárs problémát más feladatokkal, illetve területekkel, másrészről pedig a problémakör jelentőségét hangsúlyozzuk.

A szobatársak meghatározása mint klaszterezési feladat ismereteink szerint egyedül Feo és szerzőtársai (1992) tanulmányában jelent meg. Erre a szerzők a k-particionálás probléma maximalizálási verzójának egyik lehetséges alkalmazásaként hivatkoznak. A feladat háromfős szobák meghatározása, ahol az élsúlyok az egyének mint szobatársak közötti kompatibilitást méri (minél nagyobb, annál jobb). A cél pedig a k darab 3-fős klaszteren belüli élsúlyok összegének maximalizálása.

Az irodalomban ezen kívül számos egyéb, az egyenletes klaszterezéshez kapcsolódó probléma található. Szigorúbb értelemben kapcsolódó problémáknak nevezzük azokat, amelyek során a cél egyenlő méretű klaszterek meghatározása azon esetektől eltekintve, amikor az elemek száma a csoportok számának nem egész többszöröse, és így esetenként 1-es eltérés mutatkozhat a kialakított csoportok méretében.

Gyengébb értelemben kapcsolódó problémáknak tekintjük az előzőekkel együtt azokat is, amelyek megfogalmazásában a csoportok elemszámára egy megadott alsó és/vagy felső korlát vonatkozik. Ezek olyan optimalizálási problémákat adnak meg, amelyek a szigorú értelemben vett egyenletes klaszterezésnél általánosabbak, és az egyenlő méretű klaszterek előállítása csupán speciális esetként adódik.

2.1. Tanulói, szakmai és egyéb csoportok kialakítása

Tanulói- és munkacsoportok létrehozása

Az egyenletes klaszterezési feladatok egyik leggyakoribb alkalmazása az akadémiai kontextusban jelenik meg, tanulói csoportok létrehozása során. (Gallego és szerzőtársai, 2013)

A *legváltozatosabb tanulói munkacsoport problémát* (maximum diversity workgroup problem) először Weitz és Jelassi (1992) írta le. Tanulmányukban egy több kritériumot (nemzetiség, anyanyelv, nem, életkor, elvégzett egyetem, egyetemi szak, korábbi munkatapasztalat területe) is figyelembe vevő, heurisztikus alapú, döntéstámogató rendszer fejlesztését és leírását részletezik, amely az INSEAD-en (European Institute of Business Administration) került alkalmazásra. A cél a hallgatók csoportosítása projektmunkákra, és ehhez a lehető legváltozatosabb hallgatói csoportok megkonstruálása. Később a New York Egyetemen (NYU) lévő Stern School of Business is egy ehhez hasonlót vezetett be. (Weitz és Lakshminarayanan, 1996, 1998)

Mingers és O'Brien (1995) egymáshoz hasonló tanulói csoportok létrehozására helyezte a hangsúlyt, vagyis hogy a hallgatók tulajdonságait a lehető legegyenletesebben ossza el a csoportok között. A csoportok közötti különbségek minimalizálásának célja maga után vonja a csoportokon belüli karakterisztikák különbségének maximalizálását, így ez az előzővel ekvivalens probléma. (Weitz és Lakshminarayanan, 1998)

A tanulói környezetben az előző célokkal éppen ellenkezőleg az is előfordulhat, hogy hallgatókat szeretnénk valamilyen képesség vagy képességek mentén homogén csoportokra osztani. Vagyis ebben az esetben olyan csoportokat keresünk, amelyeken belül az egyének hasonlóak. Ennek célja lehet például, hogy a tanulók a képességeik szerint a számukra legmegfelelőbb szintű oktatásban részesülhessenek. (Höppner és Klawonn, 2008)

Bhadury és szerzőtársai (2000) a különböző szervezeteknél egyre nagyobb mértékben jelenlévő munkaerő diverzitásra fókuszál. Ez a számos szociális, gazdasági és politikai faktor miatt kialakult jelenség a munkahelyeken lévő különösen változatos munkavállalókra utal. A cél olyan diverz csoportok létrehozása, amelyben a különböző háttérrel rendelkező emberek végül hatékonyabban dolgozhatnak együtt. (Gallego és szerzőtársai, 2013)

Vizsgabeosztás

Lotfi és Cerveny (1991) egy négy lépésből álló rendszert implementált egy nagy egyetemen a vizsgák beosztására. Ez első lépésben a vizsgákat blokknak nevezett halmazokra

osztják úgy, hogy azokon belül minimalizálják azon hallgatók számát, akiknek egy időben több vizsgájuk is lenne. Második lépésként a blokkokat vizsganapokhoz rendelik úgy, hogy azon hallgatók száma, akiknek egy nap kettő vagy annál több vizsgája lenne, minimális legyen. Harmadik lépésként a vizsganapokat és a napokon belül a blokkokat úgy rendezik, hogy minimalizálják azon hallgatók számát, akiknek egymást követő vizsgája lenne. Negyedik lépésben pedig a vizsgákat termekhez rendelik a helykihasználás maximalizálása mellett. Ezek közül a második lépés, vagyis a vizsga blokkok napokhoz rendelése az egyenletes klaszterezés minimalizálási verziójával ekvivalens. (Weitz és Lakshminarayanan, 1998)

Konferencia szekcióbeosztás

A konferenciákon jellemzően egyszerre több tudományterület is megjelenik, az előadásokat pedig előre meghatározott időtartamú blokkokba kell beosztani. Ha meghatározunk valamilyen távolságot, amellyel mérhetővé tesszük, hogy az egyes tanulmányok mennyire kapcsolódnak ugyanazokhoz a területekhez, akkor a szekcióbeosztás egy egyenletes klaszterezési feladattal lesz ekvivalens.

Szakmai bírálók csoportjainak kialakítása

Gallego és szerzőtársai (2013) a *maximálisan diverz csoportosítási problémát* (maximally diverse grouping problem, MDGP) vizsgáló tanulmányukban Hettich és Pazzani (2006) nyomán alkalmazási területként megemlítik a szakmai bírálók csoportjainak kialakítását tudományos publikációkhoz vagy kutatás-finanszírozó alapok projektértékeléseihez.

Hettich és Pazzani (2006) egy olyan applikációt fejlesztett az amerikai Nemzeti Tudományos Alap (National Science Foundation) számára, amely segíti a programigazgatók munkáját, hogy az évente több, mint 40.000 beadott pályázatot elbírálják. Az alkalmazás adatbányászati módszerekkel panelekbe csoportosítja a beadott pályázatokat, majd segíti ezek bírálókhoz rendelését.

Habár a végleges megoldás nem egy egyszerű klaszterezéssel ekvivalens eljárás lett, hanem egy annál sokkal inkább az adott problémára igazított, a tárgyalás során ez is felmerül alternatívaként. Az egyenletes klaszterezés pozitívuma, hogy némely korábbi megközelítéssel ellentétben ez garantálja a közel azonos csoportméreteket. Negatívuma

viszont, hogy nélkülöz olyan mechanizmusokat, amely összehangolná a megoldást a szervezet struktúrájával és munkamenetével. Például nincs garancia arra, hogy a kialakított csoportok megfelelnek valamely konkrét tudományos területnek. Emellett előfordulhat, hogy egyes pályázatok csoportos elbírálásakor azok témaköréből kifolyólag heterogénebb bírálókra van szükség, míg máskor egy homogén csoport kialakítása a célszerű.

Csapatok kialakítása sportversenyeken és játékokban

Sportversenyek csapatkörének kialakításakor azonos számú csapat kerül az egyes csoportokba. A cél, hogy a csoportok minél hasonlóbbak legyenek, így a feladat megfeleltethető egy egyenletes klaszterezési feladatnak.

Online számítógépes csapatjátékoknál gyakran két csapat versenyzik egymással a győzelemért, ahol a csapatok tagjait is egy algoritmus választja ki a játékra várakozó egyének közül. Ahhoz, hogy a mérkőzés ne legyen egyoldalú, a csapatoknak azonos erősségűeknek kell lenniük, tehát hasonló csoportok kialakítása szükséges.

2.2. Ipari és informatikai alkalmazások

Erőforrások allokációja

Egyenletes klaszterezésre van szükség, amikor árucikkek, munkák vagy vásárlók egy halmazát kell elosztanunk korlátozott számú erőforrás között, és az egyes erőforrásokra jutó munkamennyiségeknek közel azonosaknak kell lenniük. Ilyen az, amikor munkákat kell gépek vagy dolgozók között elosztani úgy, hogy az egyes egységekre azonos munkamennyiség háruljon, vagy hogy a hatékonyabb munkavégzés vagy a gépek konfigurálási idejének csökkentése érdekében a munkák a lehető leghasonlóbbak legyenek. További példa a telephelyek elhelyezése úgy, hogy a különböző helyszínekről érkező árucikkeknek a lehető legkevesebbet kelljen utazniuk és a sorban állás is elkerülhető legyen. (Höppner és Klawonn, 2008)

A *csoporttechnológia* (group technology) klasszifikáció problémája a munkamennyiség szétosztásához hasonló feladat. Ebben az esetben adva van a legyártandó elemek halmaza és valamennyihez a szükséges gépi felszereltség. A cél az, hogy úgy képezzünk csoportokat az elemekből, hogy a csoporton belüli elemek feldolgozási szükségletei nagyon hasonlóak

legyenek. Egy rugalmas gyártási rendszer tervezésénél ez az egyik kezdeti lépés (Chang és Wysk, 1985; King és Nakornchai, 1982; Kumar és szerzőtársai, 1986). (Feo és Khellaf, 1990)

A *szoftverszolgáltatás* (Software as a Service, SaaS) egy egyre fontosabb szolgáltatási forma a *felhőalapú számítástechnikában* (cloud computing). Az ún. *multitenancy* lehetővé teszi egyszerre nagyszámú egyedi ügyfél kiszolgálását csupán egyetlen programkódra alapozva. Ugyanakkor a *multitenant* architektúra komplexitása gyenge teljesítményhez és alacsony erőforrás-kihasználáshoz vezethet. Az egyedi kérések pedig szintén magas működési költséggel járhatnak. Su és szerzőtársai (2015) megközelítésében az *ügyefeleket* (tenant) egyenletesen osztják el az alkalmazás példányai között a hatékonyság növelése érdekében.

Nallusamy és szerzőtársai (2009) tanulmányában a *többügynökös utazó ügynök probléma* (multiple traveling salesman problem, MTSP) során a városokat klaszterezik, és minden egyes utazó ügynök egy klasztert lát el. Ekkor kívánatos lehet az utazó ügynökökhöz azonos munkamennyiséget allokálni. (Malinen és Fränti, 2014)

Ugyanezzel a feladattal találkozhattunk a koronavírus járvány idején, amikor a mentőszolgálat dolgozóinak házhoz kell mennie tesztek elvégzéséhez. Amennyiben szeretnénk az egyénekre jutó terhet egyenletesen elosztani, egy egyenletes MTSP-t kell megoldanunk.

A vezetéknélküli szenzorhálózatok (wireless sensor networks, WSN) nagyszámú, olcsó, korlátozott erőforrással rendelkező, vezetéknélküli érzékelőkből állnak, amelyek a környezetükben lévő fizikai paramétereket észlelik és monitorozzák önszervező módon (Liao és szerzőtársai, 2013). Az egyenletes klaszterezés alkalmazása csökkentheti az energiaveszteséget és növelheti a hálózat életciklusát (Lan és szerzőtársai, 2009; Liao és szerzőtársai, 2013). (Malinen és Fränti, 2014)

Kezdőklaszterek meghatározása

Az egyenletes klaszterezés segíthet jelentőségteljesebb kezdőklaszterek meghatározásában és az *outlier* klaszterek kialakulásának elkerülésében. Banerjee és Ghosh (2002, 2006) megmutatta, hogy egy kisméretű minta elegendő egy kezdeti klaszterezés előállításához, majd a további adatpontokat ezekhez rendelték hozzá az egyenletes korlátok betartása mellett. Zhong és Ghosh (2003a,b) is figyelembe vett egyenlő méretű klasztereket előíró korlátokat, és a minta klaszterekhez történő rendeléséhez egy iteratív, páros gráfon ala-

puló heurisztikát alkalmazott. A páros gráf egyik halmazában a klasztereket reprezentáló pontok, míg a másik halmazban maguk az adatpontok szerepelnek. Ezen tanulmányok azt is megmutatják, hogy ha valamilyen háttérinformációt is figyelembe veszünk, akkor az növeli a klaszterezés hatékonyságát és skálázhatóságát. (Zhu és szerzőtársai, 2010)

Adatbányászati alkalmazások (Banerjee és Ghosh, 2006)

Az adatbányászat népszerűsödésével egyidőben nőtt az érdeklődés az olyan klaszterező algoritmusok iránt is, amelyek megfelelhetnek adatbányászati alkalmazásokhoz (Jain és szerzőtársai, 1999; Han és szerzőtársai, 2001; Fasulo, 1999; Ghosh, 2003). Az adatbányászati problémáknál használt klaszterezési eljárásoknak különösen jól skálázhatóaknak kell lenniük. Emellett számos adatbányászati alkalmazás megköveteli, hogy a kialakított klaszterek (nagyjából) egyenlő méretűek vagy fontosságúak legyenek. Ezekben az esetekben tehát az egyenletes klaszterezési megközelítés nem az adathalmaz valamilyen belső tulajdonságából következik, hanem a megfelelő alkalmazási vagy üzleti igény követeli meg azt. Erre néhány konkrét példát adnak az alábbiak:

Direkt marketing (Strehl és Ghosh, 2003; Yang és Padmanabhan, 2003): A direkt marketing kampányok gyakran azzal kezdődnek, hogy a fogyasztókat közel azonos méretű vagy egyenlő becsült bevételt generáló szegmensekbe sorolják (piaci kosár elemzés, demográfia vagy valamely weblapon történő vásárlási szokás alapján), hogy ugyanakkora értékesítési csapatot, marketing költséget, stb. tudjanak rendelni valamennyi szegmenshez.

Kategória menedzsment (Nielsen Marketing Research, 1993): A kategória menedzsment egy folyamat, amely magába foglalja a termékkategóriák üzleti egységként történő menedzselését és azok vásárlói igényeknek megfelelő üzletenként eltérő testreszabását. A kategória menedzsment során az egyik alapvető eljárás, hogy a termékeket úgy csoportosítják kategóriákba, hogy azok megfeleljenek polcok vagy szintek egységeinek. Ez különösen fontos egyenletes klaszterezési alkalmazás a nagy áruházláncok számára, mint például Waltmar esetében. Egy másik kulcsfontosságú művelet az olyan fogyasztási cikkeket előállító nagyvállalatoknál, mint amilyen a Procter & Gamble is, hogy az egymással kapcsolatban lévő cikkszámokat (stock keeping unit, SKU) olyan csoportokba rendezik, hogy azok bevétel és profit szerint összehasonlíthatóak legyenek. Mindkét eljárás során fontos a csoportok időnkénti kiigazítása a különböző termékek eladási számainak szezonális különbségei, a fogyasztói trendek, stb. következtében (Gupta és Ghosh, 2001).

Dokumentumok klaszterezése (Lynch és Horton, 1999; Baeza-Yates és Ribeiro-Neto, 1999): Amikor dokumentumok egy nagyméretű gyűjteményét klaszterezik témák szerinti hierarchiák generálásához, akkor az egyenletesség nagyban hozzájárul a későbbi könnyebb böngészéshez és navigációhoz azzal, hogy elkerüli az olyan torzított hierarchiák létrehozását, ahol a hierarchia fa különböző részeinek nem kiegyensúlyozott a mélysége, vagy ahol a leveleknek megfelelő csúcsokon változó számú dokumentum található. Hasonló elvek vonatkoznak cikkek csoportosítására weboldalakon (Lynch és Horton, 1999) és tartalomszolgáltatók tervezésénél.

Egyenletes klaszterezés *Energiatudatos Szenzor Hálózatokban* (Energy Aware Sensor Networks) (Ghiasi és szerzőtársai, 2002; Gupta és Younis, 2003): Az *osztott szenzor hálózatok* (distributed sensor network, DSN) ad-hoc mobil hálózatok, amelyek limitált számítási és kommunikációs lehetőségekkel rendelkező érzékelőket tartalmaznak (Eschenauer és Gligor, 2002). A DSN-ekben a szenzorokat csoportokba osztják, és mindegyik csoportot egy érzékelő 'fej' reprezentálja olyan attribútumok alapján, mint a térbeli elhelyezkedés, a protokoll karakterisztikák, stb. Egy további kívánatos tulajdonság, amelyet gyakran külső puha korlátként írnak elő a klaszterezésre, hogy minden csoport hasonló mértékű energiát fogyasszon, mivel így a hálózat egésze megbízhatóbb és skálázhatóbb.

Nagyméretű számtógépes programok tárolása

Ha egy operációs rendszer *lapozást* (paging) valósít meg, akkor a memória allokálása *memórialaponként* (memory page) történik. Ekkor a memóriakezelő kevés szabad fizikai memória esetén háttértárra mentheti az egyes lapokat. Ezt a műveletet nevezzük lapozásnak. (Benyó és szerzőtársai, 2000)

A particionálás kérdése akkor is felmerül, amikor nagyméretű számítógépes programokat szeretnénk tárolni *lapozott memórián* (paged memory). A programok általában egymáshoz kapcsolódó eljárások halmazaként épülnek fel. A probléma lényege, hogy az alprogramokat megadott méretű lapokhoz kell rendelni úgy, hogy a különböző lapokon lévő objektumok közötti hivatkozások száma minimális legyen. Az alprogramok mérete rögzített, és ugyanannak az alprogramnak több lapon is való tárolása nem megengedett. (Feo és Khellaf, 1990)

VLSI tervezés (Feo és Khellaf, 1990)

A Feo és Khellaf (1990) által megfogalmazott gráf particionálási probléma során a feladat egy gráf csúcsainak csoportosítása egy megadott mérték optimalizálása mellet. Ennek egy lehetséges alkalmazásaként említik a *VLSI* (Very-Large-Scale Integration) tervezést. A VLSI az a folyamat, amely által komplex elektronikai áramköröket integrálnak egy szilikon lapkára egy chip készítéséhez. Az áramkör gyakran számos modulból áll, amelyek fizikai kapcsolatokon, pl. vezetéken keresztül kommunikálnak egymással. A modulok integrálásának első lépése, hogy egymáshoz képest fizikailag elhelyezzék őket. A második lépés, hogy megteremtsék vezetékek segítségével az összekapcsolást. Az elhelyezés és összekapcsolás során két alapvető célkitűzés van. Először is, hogy minimalizálják a vezeték teljes szükséges hosszát, másrészt pedig, hogy minimalizálják a chiphez szükséges szilikon lapka méretét. Az összekapcsoltság szerinti csoportosítás elősegíti mindkét kitűzött cél megvalósulását.

3. fejezet

A klaszterezési feladatok komplexitása

Az egyenletes klaszterezési feladatokra tekinthetünk kombinatorikai problémaként is, hiszen n elemet (összes hallgató száma) kell k darab (szobák száma) m nagyságú (szobák mérete) csoportba beosztani. Ennek megfelelően megadhatjuk az összes lehetséges kombináció számát az

$$\frac{\binom{n}{m} \cdot \binom{n-m}{m} \cdot \binom{n-2m}{m} \cdot \ldots \cdot \binom{m}{m}}{(n/m)!} = \frac{n!}{(n/m)!\ (m!)^{n/m}}, \tag{3.1}$$

formulával, mivel az első szobába n hallgató közül m-et kell beosztani, a második szobába $n - m$ hallgató közül további m-t és így tovább, de mivel a csoportok sorrendje nem számít, így osztanunk is kell a szobák számának faktoriálisával.

Ez alapján felmerülhet ötletként, hogy az összes lehetséges eset végignézésével egyértelműen meg tudjuk határozni a feladat szerinti optimumot. Ugyanakkor, ha egyenként vennék sorra minden egyes eshetőséget, akkor még viszonylag kis n és m értékek mellett is nagyon sok idő lenne, mire végeznénk.

A (3.1) formula értékét relatíve kis k (szobák száma) és m paraméterek esetén a 3.1-es táblázat szemlélteti. Látható, hogy adott m esetén a $k = n/m$ növekedésével a hányados rendkívül gyorsan nő, így az összes eset számbavétele nem járható út. Olyannyira nem, hogy például 9 szoba és 4-fős szobák (tehát összesen csupán 36 fő) esetén, ha másodpercenként egybillió (10^{12}) esetet tudnánk végignézni, még akkor is több, mint tízezer év ($3{,}156 \cdot 10^{11}$ másodperc) kellene ahhoz, hogy minden lehetőséget megvizsgáljunk. Így sejthetjük, hogy kifejezetten komplex problémákkal állunk szemben. Ha szeretnénk a komplexitást precízebben megragadni, akkor komolyabb eszközökre, vagyis a bonyolultságelmélet fogalmaira van szükségünk.

k	m = 2	m = 3	m = 4	m = 10	m = 20
2	3	10	35	92378	6,89E+10
3	15	280	5775	9,25E+11	9,63E+25
4	105	15400	2627625	1,96E+20	8,51E+43
5	945	1401400	2,55E+09	4,03E+29	9,12E+63
6	10395	1,91E+08	4,51E+12	5,06E+39	4,48E+85
7	135135	3,62E+10	1,32E+16	2,87E+50	5,29E+108
8	2027025	9,16E+12	5,93E+19	5,90E+61	9,53E+132
9	3,45E+07	2,98E+15	3,88E+23	3,75E+73	1,85E+158

3.1. táblázat. Az összes lehetséges rendezés száma különböző k (csoportok száma) és m (szobatársak száma) értékek esetén.

A fejezet hátralévő részében az m-szobatárs probléma, valamint általánosan a klaszterezési problémák bonyolultságelméleti szempontból történő vizsgálatához elsőként bevezetjük a kapcsolódó definíciókat és eredményeket, majd az irodalomban tárgyalt klaszterezési feladatok nehézségi eredményeit tárgyaljuk.

3.1. Bonyolultságelméleti alapok

Az optimalizálási problémák komplexitásának vizsgálatához a bonyolultásgelméleti alapfogalmakat és eredményeket általánosan Ausiello és szerzőtársai (2003) alapján vezetjük be. Ahol ettől eltérünk, azt adott esetben jelezzük. A leírás során kifejezett célunk, hogy a lehetőségekhez képest átfogóan mutassuk be az egymásra épülő, később releváns fogalmakat, ezzel a disszertáció keretein belül megadva a kellő elméleti hátteret a bonyolultságelméleti eredmények értelmezéséhez. Az optimalizálási problémák definíciói és eredményei az eldöntési problémák fogalmaira építenek, így elsőként ezeket adjuk meg.

3.1.1. Eldöntési problémák osztályai

Eldöntési probléma

A problémák, amiket meg szeretnénk oldani, elég különfélék lehetnek. Általánosságban ezek kifejezhetőek egy $P \subseteq I \times \mathcal{S}$ relációval, ahol I a *probléma eseteinek* vagy *példányainak* (problem instances) halmaza és $\mathcal{S}$ a *probléma megoldásainak* (problem solutions) halmaza.

Alternatívaként tekinthetünk egy $p(x, y)$ állítást is, amely akkor és csak akkor igaz, ha $(x, y) \in P$.

A bonyolultságelmélet egyik meghatározó eleme az eldöntési probléma. Az eldöntési problémák során csupán arra vagyunk kíváncsiak, hogy a probléma egy esetére egy adott kritérium teljesül-e, így a P reláció egy $f : I \to \mathcal{S}$ függvényre egyszerűsödik, ahol $\mathcal{S} = \{\text{IGEN}, \text{NEM}\}$.[1]

3.1.1. Definíció. Eldöntési probléma.
Egy $\mathcal{P}$ problémát eldöntési problémának *(decision problem) nevezünk, ha a $\mathcal{P}$ probléma összes esetének $I_\mathcal{P}$ halmaza particionálható* pozitív esetek *(positive instances) $Y_\mathcal{P}$ és* negatív esetek *(negative instances) $N_\mathcal{P}$ halmazára, és a probléma azt kéri, hogy döntsük el egy $x \in I_\mathcal{P}$ esetre, hogy $x \in Y_\mathcal{P}$ teljesül-e.*

Arora és Barak (2009) nyomán bemutatjuk a bonyolultságelmélet egyik legalapvetőbb eldöntési problémáját, a SAT-ot, amely a propozicionális logikából jön (propozicionális logikáról lásd Simon (n.d.) "Matematikai logika" c. jegyzetét).

3.1.2. Példa. SAT, 3SAT
Egy *Boole-formula* (Boolean formula) az $u_1, \ldots, u_n$ változókon magukból a változókból, valamint az AND ($\wedge$, "és"), OR ($\vee$, "vagy") és NOT ($\neg$, "negált") logikai operátorokból áll. Például $(u_1 \vee u_2) \wedge (u_2 \vee u_3) \wedge (u_3 \vee u_1)$ egy Boole-formula. Ha φ egy Boole-formula az $u_1, \ldots, u_n$ változók és $z \in \{0, 1\}^n$ felett, akkor $\varphi(z)$ jelöli φ értékét, amikor φ változói z értékeihez vannak rendelve, ahol az 1-et IGAZ-ként (TRUE), míg a 0-t HAMIS-ként (FALSE) értelmezzük. Egy φ formula *(logikailag) kielégíthető* (satisfiable), ha létezik olyan z hozzárendelés, amelyre $\varphi(z)$ értéke IGAZ. Ellenkező esetben φ *nem (logikailag) kielégíthető* (unsatisfiable). A fenti példa kielégíthető, ha u_1, u_2, u_3 változók közül legalább kettőhöz 1 értéket rendelünk, pl. $u_1 = 1, u_2 = 1$ és $u_3 = 0$.

Egy Boole-formula az $u_1, \ldots, u_n$ változók felett *konjunktív normál formában* (conjunctive normal form, CNF) van megadva, hogyha a változók és azok negáltjainak OR-jainak AND-jeiként van kifejezve, vagyis általánosan az

$$\wedge_i \left(\vee_j v_{ij}\right)$$

alakban van megadva, ahol v_{ij} egy u_k változó vagy annak $\neg u_k$ negáltja. A formulában a

[1] Eldöntési problémákról, főként gráfelméleti vonatkozásban, lásd még Katona és szerzőtársai (2006).

v_{ij} tagokat *literálisoknak* (literal) nevezzük[2], a $(\vee_j v_{ij})$ tagokat pedig *klózoknak* (clause). Egy kCNF egy olyan CNF, ahol bármely klóz legfeljebb k literálist tartalmaz. Ekkor a SATISFIABILITY (röviden SAT) eldöntési probléma a következő:

3.1.3. Eldöntési probléma. *SAT*

Input: *Egy C CNF.*

Kérdés: *Van-e C-t kielégítő hozzárendelés?*

Ekkor a SAT probéma egy I_{SAT}-beli esete egy C CNF, amelyre $C \in Y_{SAT}$, ha kielégíthető, ellenkező esetben pedig $C \in N_{SAT}$. Az előzőhöz hasonlóan, 3CNF formulákra vonatkozóan felírhatjuk a 3SAT eldöntési problémát.

3.1.4. Eldöntési probléma. *3SAT*

Input: *Egy C 3CNF.*

Kérdés: *Van-e C-t kielégítő hozzárendelés?*

P osztály

A számítási bonyolultság elméletének fő célja, hogy a számításhoz szükséges erőforrás (idő, hely) szempontjából karakterizálja a problémák csoportjait. Ezeket a csoportokat nevezzük bonyolultsági osztályoknak. A bonyolultság megközelítéséhez a számítási időt vesszük alapul, amely függ az input méretétől. Feltételezzük, hogy létezik egy *kódolási séma* (encoding scheme), amellyel bármely probléma bármely esete felírható karakterek egy sorozataként valamely ábécé fölött. A továbbiakban egy probléma x esetére ugyanazzal az x szimbólummal jelöljük azt a karaktersorozatot is, amely az x eset kódolásából ered. Jelölje $|x|$ a kódolás során kapott karaktersorozat hosszát.

Az algoritmusok futásidejének méréséhez az *egységes költség* (uniform cost) modellt alkalmazzuk, amelynek alapja, hogy meghatározzuk az algoritmus terminálása előtt elvégzett összes művelet számát.[3] Az algoritmusok futásidejét az aszimptotikus viselkedésüknek megfelelően definiáljuk, ahol a függvény változója az input mérete. Jelölje $\hat{t}_{\mathcal{A}}(x)$ az $\mathcal{A}$ algoritmus futásidejét az x inputon. Ekkor $\mathcal{A}$ *futásideje a legrosszabb esetben* (worst

[2]Így a CNF literálisok diszjunkcióinak konjunkciója. (Simon, n.d., "Matematikai logika" c. jegyzet)

[3]Az egységes költség modellel implicit feltételezzük, hogy egy művelet bármely inputon konstans időben elvégezhető annak méretétől függetlenül. Ez a mi tárgyalásunkhoz megfelelő, ugyanakkor egyes esetekben komoly problémákat okozhat. Alternatívaként alkalmazható a *logaritmikus költség* modell, részletekért lásd Ausiello és szerzőtársai (2003).

case running time) legyen

$$t_{\mathcal{A}}(n) = \max\left\{\hat{t}_{\mathcal{A}}(x) \;\middle|\; \forall x : |x| \leq n\right\}.$$

3.1.5. Definíció. Algoritmus bonyolultságának korlátai.

Azt mondjuk, hogy az $\mathcal{A}$ algoritmus

1. *bonyolultsága (vagy bonyolultságának felső korlátja) $\mathcal{O}(g(n))$, hogyha $t_{\mathcal{A}}(n)$ $\mathcal{O}(g(n))$ nagyságrendű.*[4]
2. *bonyolultsága (vagy bonyolultságának alsó korlátja) $\Omega(g(n))$, hogyha $t_{\mathcal{A}}(n)$ $\Omega(g(n))$ nagyságrendű.*
3. *bonyolultsága (pontosan) $\Theta(g(n))$, hogyha $t_{\mathcal{A}}(n)$ $\Theta(g(n))$ nagyságrendű.*

Azt, hogy egy eldöntési probléma milyen futásidőben oldható meg, az azt megoldó algoritmusok futásidején keresztül adjuk meg. Ennek segítségével pedig a problémák különböző osztályait definiálhatjuk.

3.1.6. Definíció. Probléma megoldása.

Egy $\mathcal{A}$ algoritmus megold egy $\mathcal{P}$ eldöntési problémát, ha az algoritmus terminál (vagyis az algoritmus futása megáll) minden $x \in I_{\mathcal{P}}$ esetre, és a visszatérési érték IGEN *akkor és csak akkor, hogyha $x \in Y_{\mathcal{P}}$. Ekkor azt mondjuk, hogy az $\mathcal{A}$ algoritmus* felismeri *(recognize) az $Y_{\mathcal{P}}$ halmazt. Továbbá azt mondjuk, hogy a $\mathcal{P}$ problémát $t(n)$ időben oldottuk meg, hogyha az $\mathcal{A}$ algoritmus időbeli bonyolultsága $t(n)$.*

3.1.7. Definíció. TIME($f(n)$) bonyolultsági osztály.

Bármely $f(n)$ függvényre jelölje TIME($f(n)$) *azon eldöntési problémák osztályát, amelyek megoldhatóak $\mathcal{O}(f(n))$ időn belül.*

3.1.8. Definíció. P osztály.

Legyen P$= \cup_{k=0}^{\infty}$TIME(n^k) *az összes olyan eldöntési probléma osztálya, amelyek az input méretének egy polinomiális függvényével arányos időben megoldhatóak.*

A P osztályba tehát azon eldöntési problémák tartoznak, amelyek megoldására létezik polinomiális futásidejű algoritmus. Általánosan ezeket a problémákat tekintjük hatékonyan megoldhatónak.

[4] $\mathcal{O}(g(n)), \Omega(g(n))$ és $\Theta(g(n))$ az algoritmus aszimptotikus viselkedését írja le. Az, hogy $f(n)$ $\mathcal{O}(g(n))$ nagyságrendű , vagy másképpen $f(n) = \mathcal{O}(g(n))$, azt jelenti, hogy $\exists k > 0$ $\exists n_0$ $\forall n > n_0 : |f(n)| \leq kg(n)$. Hasonlóan, $f(n) = \Omega(g(n))$ azt jelenti, hogy $\exists k > 0 \exists n_0 \forall n > n_0 : f(n) \geq kg(n)$. Végül $f(n) = \Theta(g(n))$ akkor, hogyha $f(n) = \mathcal{O}(g(n))$ és $f(n) = \Omega(g(n))$ egyszerre fennáll.

NP osztály

Annak a költsége, hogy ellenőrizzük a problémák megoldásait, további szempontként szolgálhat a problémák bonyolultságának karakterizálásához. Erre a célra lett bevezetve a *nemdeterminisztikus algoritmus* koncepciója, amely olyan lépéseket is végrehajthat, amelyben az algoritmus több állapot közül *választ* (guess) egyet. Amíg a korábbi (determinisztikus) algoritmusok által elvégzett számítások struktúrája lineáris volt, a választásoknak köszönhetően a nemdeterminisztikus számítások egy fához hasonlító struktúrával írhatóak le. Ezt a struktúrát hívjuk *számítási fának* (computational tree), amelyben a választási pontok felelnek meg az elágazásoknak.

3.1.9. Definíció. Nemdeterminisztikus problémamegoldás.
Legyen adott egy $\mathcal{P}$ eldöntési probléma. Egy $\mathcal{A}$ nemdeterminisztikus algoritmus megoldja *$\mathcal{P}$-t, hogyha bármely $x \in I_{\mathcal{P}}$ esetre $\mathcal{A}$ megáll a választások bármely lehetséges sorozatára, és $x \in Y_{\mathcal{P}}$ akkor és csak akkor, ha van a választásoknak legalább egy olyan sorozata, amelynek megfelelően az algoritmus visszatérési értéke* IGEN.

3.1.10. Definíció. Nemdeterminisztikus algoritmus bonyolultsága.
Egy $\mathcal{A}$ nemdeterminisztikus algoritmus egy $\mathcal{P}$ problémát $t(n)$ időben old meg, hogyha bármely $x \in I_{\mathcal{P}}$ esetre, amelyre $|x| = n$, $\mathcal{A}$ megáll a választások bármely lehetséges sorozatára, és $x \in Y_{\mathcal{P}}$ akkor és csak akkor, hogyha létezik a választásoknak legalább egy olyan sorozata, amelynek megfelelően az algoritmus visszatérési értéke legfeljebb $t(n)$ időben IGEN.

Az, hogy van a választásoknak olyan sorozata, amelyre az algoritmus visszatérési értéke $t(n)$ időben IGEN, pontosan annak felel meg, hogy egy feltételezett megoldásról $t(n)$ időben ellenőrizhető, hogy az valóban megoldás-e. A nemdeterminisztikus algoritmusok révén a következő osztályokat definiálhatjuk.

3.1.11. Definíció. Nemdeterminisztikus bonyolultsági osztályok.
Bármely $f(n)$ függvényre jelölje NTIME$f(n)$ *azon eldöntési problémák csoportját, amelyek megoldhatóak egy nemdeterminisztikus algoritmus által $\mathcal{O}(f(n))$ időben.*

3.1.12. Definíció. NP osztály.
Legyen NP$= \cup_{k=0}^{\infty}$ NTIME(n^k) *az összes olyan eldöntési probléma osztálya, amelyek az input méretének egy polinomiális függvényével arányos időben megoldhatóak egy nemdeterminisztikus algoritmus által.*

Mivel egy hagyományos (determinisztikus) algoritmus egy nemdetermnisztikus speciális esete, amelyben nincs véletlen választás, ezért P $\subseteq$ NP rögtön következik. Az, hogy P = NP teljesül-e a bonyolultságelmélet egyik mai napig nem megválaszolt, központi kérdése. Általánosan úgy gondoljuk, hogy P $\neq$ NP, viszont amíg nem nyer bizonyítást, addig számos eredménynél feltételként meg kell említenünk.

NP-teljesség

A különböző problémák bonyolultságai között kapcsolatot teremthetünk redukciók (visszavezetések, angolul 'reduction') alkalmazásával. Alapvetően egy redukció egy $\mathcal{P}_1$ problémáról egy $\mathcal{P}_2$ problémára egy módszer arra, hogy megoldjuk $\mathcal{P}_1$-et egy $\mathcal{P}_2$-re vonatkozó algoritmus által. Ekkor a $\mathcal{P}_1$ és $\mathcal{P}_2$ problémák viszonyára gondolhatunk úgy, hogy $\mathcal{P}_2$ legalább olyan nehéz, mint $\mathcal{P}_1$ (feltéve, hogy a redukció csak 'egyszerű' számításokat igényel). Ugyanez precízebben, ha $\mathcal{P}_2$-t hatékonyan, például polinomiális időben meg tudjuk oldani, akkor ez teljesül $\mathcal{P}_1$-re is, vagyis $\mathcal{P}_2 \in P$-ből $\mathcal{P}_1 \in P$ következik. Ugyanakkor, ha $\mathcal{P}_1$-et bizonyítottan nem tudjuk megoldani hatékonyan, akkor biztosan nem tudjuk megoldani hatékonyan $\mathcal{P}_2$-t sem, vagyis $\mathcal{P}_1 \notin P$ maga után vonja $\mathcal{P}_2 \notin P$-t.

Annak megfelelően, hogy milyen feltételezéssel élünk azt illetően, hogy a $\mathcal{P}_2$ egy megoldása hogyan alkalmazható a $\mathcal{P}_1$ megoldására, különböző redukciókat definiálhatunk.

3.1.13. Definíció. Karp-redukció.
Azt mondjuk, hogy egy $\mathcal{P}_1$ probléma Karp-redukálható *vagy* visszavezethető *(Karp-reducible) egy $\mathcal{P}_2$ problémára, ha létezik egy $\mathcal{R}$ algoritmus, amely ha adott a $\mathcal{P}_1$ probléma valamely $x \in I_{\mathcal{P}_1}$ esete, akkor azt a $\mathcal{P}_2$ probléma egy $y \in I_{\mathcal{P}_2}$ esetévé transzformálja úgy, hogy $x \in Y_{\mathcal{P}_1}$ akkor és csak akkor, ha $y \in Y_{\mathcal{P}_2}$. Ekkor azt mondjuk, hogy $\mathcal{R}$ egy* Karp-redukció *$\mathcal{P}_1$-ről $\mathcal{P}_2$-re, és ezt $\mathcal{P}_1 \leq_m \mathcal{P}_2$-vel jelöljük. Ha $\mathcal{P}_1 \leq_m \mathcal{P}_2$ és $\mathcal{P}_2 \leq_m \mathcal{P}_1$ egyszerre teljesül, akkor azt mondjuk, hogy $\mathcal{P}_1$ és $\mathcal{P}_2$* Karp-ekvivalens*, jelölésében $\mathcal{P}_1 \equiv_m \mathcal{P}_2$.*

3.1.14. Definíció. Polinomiális idejű Karp-redukció.
Azt mondjuk, hogy egy $\mathcal{P}_1$ probléma polinomiálisan Karp-redukálható *vagy* polinomiálisan visszavezethető *egy $\mathcal{P}_2$ problémára akkor és csak akkor, hogyha $\mathcal{P}_1$ Karp-redukálható $\mathcal{P}_2$-re, és a megfelelő $\mathcal{R}$ redukció polinomiális futásidejű algoritmus. Ezt $\mathcal{P}_1 \leq_m^p \mathcal{P}_2$-vel jelöljük, és amennyiben $\mathcal{P}_1 \leq_m^p \mathcal{P}_2$ és $\mathcal{P}_2 \leq_m^p \mathcal{P}_1$ egyszerre teljesül, úgy $\mathcal{P}_1 \equiv_m^p \mathcal{P}_2$.*

A következő definíciók segítségével megadjuk azon problémák részosztályát, amelyeket

a legnehezebbnek gondolunk egy adott osztályon belül (egy megfelelő redukcióra vonatkozóan).

3.1.15. Definíció. $\mathcal{C}$-teljes.
Bármely $\mathcal{C}$ bonyolultsági osztály esetén azt mondjuk, hogy egy $\mathcal{P} \in \mathcal{C}$ eldöntési probléma teljes *(complete) $\mathcal{C}$-ben, vagy ekvivalensen* $\mathcal{C}$-teljes *egy $\leq_r$ redukcióra vonatkozóan, ha bármely másik $\mathcal{P}_1 \in \mathcal{C}$ problémára $\mathcal{P}_1 \leq_r \mathcal{P}$.*

3.1.16. Definíció. NP-teljes.
Azt mondjuk, hogy egy $\mathcal{P}$ eldöntési probléma NP-teljes *(NP-complete), hogyha teljes NP-ben $\leq_m^p$-re vonatkozóan, vagyis $\mathcal{P} \in NP$, és bármely $\mathcal{P}_1 \in NP$ eldöntési problémára $\mathcal{P}_1 \leq_m^p \mathcal{P}$.*

Az NP osztályon belül az NP-teljes problémákat gondoljuk a legnehezebbeknek. Ha egy problémáról belátjuk, hogy az NP-teljes, akkor arra úgy tekintünk, hogy nem tudjuk hatékonyan megoldani. A SAT-ra, illetve 3SAT-ra vonatkozó NP-teljességi eredményt Arora és Barak (2009) nyomán a következő tétel mondja ki.

3.1.17. Tétel. Cook-Levin Tétel (Cook (1971), Levin (1973))
1. A SAT NP-teljes.
2. A 3SAT NP-teljes.

3.1.2. Optimalizálási problémák osztályai

NPO osztály

A bonyolultságelmélet alapvető definícióinak többsége eldöntési problémákra lett megadva. Ahhoz, hogy optimalizálási problémákkal foglalkozzunk, új fogalmak bevezetése szükséges. Emellett az optimalizálási problémák bonyolultságának és az eldöntési problémák bonyolultságának kapcsolatát is tárgyalnunk kell. Az optimalizálási probléma formális definíciója az alábbi.

3.1.18. Definíció. Optimalizálási probléma.
Egy $\mathcal{P}$ optimalizálási problémát *(optimization problem) az $(I_\mathcal{P}, SOL_\mathcal{P}, m_\mathcal{P}, goal_\mathcal{P})$ objektumnégyes karakterziál, ahol*

1. *$I_\mathcal{P}$ a $\mathcal{P}$ eseteinek halmaza;*
2. *$SOL_\mathcal{P}$ egy függvény, amely bármely $x \in I_\mathcal{P}$ input esethez az x-hez tartozó* megengedett megoldások *(feasible solutions) halmazát rendeli;*
3. *$m_\mathcal{P}$ egy (x, y) $x \in I_\mathcal{P}, y \in SOL_\mathcal{P}(x)$ párokon definiált* mértékfüggvény *(measure function). Valamennyi (x, y) párra $m_\mathcal{P}(x, y)$ egy pozitív egész értéket ad meg, amely az y megendedett megoldás értékét adja;*
4. *$goal_\mathcal{P} \in \{MIN, MAX\}$ adja meg, hogy $\mathcal{P}$ maximalizálási vagy minimalizálási probléma.*

Egy x input esetre jelöle $\mathrm{SOL}^*_\mathcal{P}(x)$ az x *optimális megoldásainak* (optimal solutions) halmazát. Az x eset egy $y^*(x)$ optimális megoldásának értékét jelölje $m^*_\mathcal{P}(x)$. Az optimalizálási problémákra az NP osztállyal analóg bonyolultsági osztály az NPO.

3.1.19. Definíció. NPO osztály.
Egy $\mathcal{P} = (I, SOL, m, goal)$ optimalizálási probléma az NPO osztályba *tartozik, hogyha a következőek teljesülnek:*

1. *Az esetek I halmaza polinomiális időben felismerhető;*
2. *Létezik egy q polinom, amelyre egy adott $x \in I$ esetén bármely $y \in SOL(x)$ megengedett megoldásra $|y| \leq q(|x|)$, és emellett bármely y-ról, amelyre $|y| \leq q(|x|)$, polinomiális időben eldönthető, hogy $y \in SOL(x)$ teljesül-e;*
3. *Az m mértékfüggvény polinomiális időben számolható.*

A $\mathcal{P}$ optimalizálási probléma megoldásának megközelítésétől függően több kapcsolódó problémát is definiálhatunk, melyek közül számunkra a megfogalmazható eldöntési probléma releváns (ezen kívül definiálható még konstruktív és kiértékelési probléma).
Eldöntési probléma ($\mathcal{P}_D$) - Adott $x \in I$ eset és $K \in \mathbb{Z}^+$ pozitív egész esetén döntsük el, hogy $m^*(x) \geq K$ (ha goal = MAX) vagy $m^*(x) \leq K$ (ha goal = MIN). Ha goal = MAX, az $\{(x, K) | x \in I \wedge m^*(x) \geq K\}$ (vagy goal = MIN esetén az $\{(x, K) | x \in I \wedge m^*(x) \leq K\}$) halmazt a $\mathcal{P}$ *alapjául szolgáló nyelvnek* (underlying language) nevezzük.

Az alábbi eredmény mutatja, hogy az NPO osztály az NP osztály természetesen adódó optimalizálási megfelelője.

3.1.20. Tétel. *Bármely $\mathcal{P} \in NPO$ optimalizálási problémához tartozó $\mathcal{P}_D$ eldöntési probléma NP-beli.*

NP-nehéz optimalizálási probléma

Az eldöntési problémák relatív bonyolultságát a polinomiális idejű Karp-redukción, és az ehhez kapcsolódó NP-teljességen keresztül ragadtuk meg. Ez azonban nem alkalmazható optimalizálási problémákra, így más eszközre van szükségünk. Ez lesz a polinomiális idejű Turing-redukció.

3.1.21. Definíció. Orákulum.
Legyen $\mathcal{P}$ egy (akár többváltozós) $f : I_{\mathcal{P}} \to \mathcal{S}_{\mathcal{P}}$ függvény kiszámításának problémája. Egy orákulum *(oracle) a $\mathcal{P}$ problémához egy olyan absztrakt szerkezet, amely bármely $x \in I_{\mathcal{P}}$ esetre megadja az $f(x) \in \mathcal{S}_{\mathcal{P}}$ értéket. Feltételezzük, hogy mindezt az orákulum akár egy számítási lépésben teszi meg.*

3.1.22. Definíció. Turing-redukció.
Legyen $\mathcal{P}_1$ egy (akár többváltozós) $g : I_{\mathcal{P}_1} \to \mathcal{S}_{\mathcal{P}_1}$ függvény kiszámításának problémája. Azt mondjuk, hogy $\mathcal{P}_1$ Turing-redukálható *(Turing-reducible) egy $\mathcal{P}_2$ problémára, ha létezik egy $\mathcal{R}$ algoritmus $\mathcal{P}_1$-re, amely egy $\mathcal{P}_2$ problémához tartozó orákulumot használ. Ekkor azt mondjuk, hogy az $\mathcal{R}$ egy* Turing-redukció *$\mathcal{P}_1$-ről $\mathcal{P}_2$-re, és ezt $\mathcal{P}_1 \leq_T \mathcal{P}_2$-vel jelöljük. Amennyiben $\mathcal{P}_1 \leq_T \mathcal{P}_2$ és $\mathcal{P}_2 \leq_T \mathcal{P}_1$ egyszerre teljesül, $\mathcal{P}_1$ és $\mathcal{P}_2$ Turing-ekvivalensek, amelyet $\mathcal{P}_1 \equiv_T \mathcal{P}_2$-vel jelölünk.*

A Karp-redukció a Turing-redukció egy speciális esete. Ahogy a Karp-redukciónál, itt is bevezethetünk $\leq_T^p$ *polinomiális idejű Turing-redukciót*, amelyre $\mathcal{P}_1 \leq_T^p \mathcal{P}_2$ akkor és csak akkor, hogyha $\mathcal{P}_1 \leq_T \mathcal{P}_2$, és az $\mathcal{R}$ Turing-redukció az input mérete szerint polinomiális időben számolható.

3.1.23. Definíció. NP-nehéz probléma.
Egy $\mathcal{P}$ optimalizálási problémát NP-nehéznek *(NP-hard) nevezünk, hogyha bármely $\mathcal{P}' \in$ NP eldöntési problémára $\mathcal{P}' \leq_T^p SOL_{\mathcal{P}}^*$, vagyis $\mathcal{P}'$ polinomiális időben megoldható egy olyan algoritmussal, amely egy olyan orákulumot használ, amely minden $x \in I_{\mathcal{P}}$ esetre megadja az x optimális $y^*(x)$ megoldását annak $m_{\mathcal{P}}^*(x)$ értékével együtt.*

Vagyis egy probléma NP-nehéz, ha legalább olyan nehéz megoldani (idő szerinti bonyolultságot tekintve és a polinomiális idejű redukciót nem számítva), mint bármely NP-beli problémát.

3.1.24. Tétel. *Legyen adott egy $\mathcal{P} \in NPO$ optimalizálási probléma. Ha a $\mathcal{P}$ alapjául szolgáló nyelv NP-teljes, akkor $\mathcal{P}$ NP-nehéz.*

Így ha egy NPO-beli $\mathcal{P}$ problémáról belátjuk, hogy NP-nehéz, akkor azzal $\mathcal{P}$ egyből az NPO osztály legbonyolultabb problémáinak szintjére kerül.

A szakirodalomban találkozhatunk az *NP-teljes optimalizálási probléma* elnevezéssel is. Ez alatt olyan optimalizálási problémákat értenek, amelyek az eldöntési probléma verziójukban teljesek, vagyis a 3.1.24. Tétel értelmében a 3.1.23. Definíció szerinti NP-nehéz optimalizálási problémák.

Approximációs osztályok

Az NP-nehéz optimalizálási problémákat a természetükből adódó bonyolultságuk miatt nem tudjuk egy konkrét módon, hatékonyan, vagyis polinomiális időben megoldani (amennyiben P $\neq$ NP). A megoldás közelítésére nyújtanak eszközt az approximációs algoritmusok. Ezek polinomiális futásidőben határoznak meg egy olyan megengedett megoldást, amely valamilyen előre ismert mértékben garantáltan megközelíti az optimumot.

3.1.25. Definíció. Approximációs algoritmus, közelítő megoldás.
Legyen adott egy $\mathcal{P} = (I, SOL, m, goal)$ optimalizálási probléma. Azt mondjuk, hogy egy $\mathcal{A}$ algoritmus approximációs algoritmus *(approximation algorithm) $\mathcal{P}$-re, hogyha az bármely $x \in I$ esetre visszaad egy* közelítő megoldást*, vagyis $\mathcal{A}(x) \in SOL(x)$ megengedett megoldást.*

Az NP-nehéz optimalizálási problémákat különböző approximációs osztályokba soroljuk aszerint, hogy milyen approximálhatósági tulajdonságokkal rendelkeznek. Az első ilyen az APX osztály, amelynek különösen fontos szerepe van az NPO osztályon belül. Ha egy NP-nehéz optimalizálási problémáról belátjuk, hogy APX-beli, az azt jelenti, hogy a konkrét megoldás megtalálásának nehézsége ellenére a megoldás valahogyan mégis megközelíthető.

3.1.26. Definíció. Teljesítményi arány.
Legyen adott egy $\mathcal{P}$ optimalizálási probléma. Ekkor a $\mathcal{P}$ bármely x esetére és az x bármely y megengedett megoldásának x-re vonatkozó teljesítményi aránya *(performance ratio) legyen*

$$R(x, y) = \max\left(\frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)}\right).$$

3.1.27. Definíció. r-approximációs algoritmus.
Legyen adott egy $\mathcal{P}$ optimalizálási probléma, és egy $\mathcal{A}$ approximációs algoritmus $\mathcal{P}$-re. Azt mondjuk, hogy $\mathcal{A}$ egy r-approximációs algoritmus *(r-approximate algorithm) $\mathcal{P}$-re, hogyha a $\mathcal{P}$ bármely x esetére az $\mathcal{A}(x)$ approximációs megoldás teljesítményi aránya felülről korlátos r-rel:*

$$R(x, \mathcal{A}(x)) \leq r.$$

3.1.28. Definíció. APX osztály.
Az APX *azon NPO-beli $\mathcal{P}$ problémák osztálya, amelyekre valamely $r \geq 1$ értékre létezik polinomiális futásidejű r-approximációs algoritmus $\mathcal{P}$-re.*

Vannak optimalizálási problémák, amelyekre a megoldást egy r-approximációnál valamilyen értelemben erősebb módon tudjuk közelíteni. Ezekben az esetekben vállalva a nagyobb számítási szükségletet, tetszőlegesen közel kerülhetünk a megoldáshoz.

3.1.29. Definíció. Polinomiális futásidejű approximációs séma (PTAS).
Legyen $\mathcal{P}$ egy NPO-beli probléma. Azt mondjuk, hogy egy $\mathcal{A}$ algoritmus polinomiális futásidejű approximációs séma *(polynomial-time approximation scheme, PTAS) $\mathcal{P}$-re, hogyha a $\mathcal{P}$ bármely x esetére és bármely $r > 1$ racionális számra az $\mathcal{A}$ az (x, r) inputra alkalmazva visszaadja az x egy r-approximációs megoldását $|x|$-ben polinomiális időn belül.*

Habár $|x|$-ben mindig polinomiális, egy PTAS futásideje függhet $1/(r-1)$-től, ugyanis minél jobb az approximáció, annál nagyobb a futásidő.

3.1.30. Definíció. PTAS osztály.
A PTAS *azon NPO-beli problémák osztálya, amelyekre van polinomiális idejű approximációs séma.*

Az előzőnél erősebb megszorítást, viszont az alkalmazhatóság szempontjából egyúttal sokkal jobb algoritmusokat jelent, hogyha az approximációs algoritmus futásideje az input mérete mellett a teljesítményi arány inverzében is polinomiális.

3.1.31. Definíció. Teljesen polinomiális idejű approximációs séma (FPTAS).
Legyen $\mathcal{P}$ egy NPO-beli probléma. Azt mondjuk, hogy egy $\mathcal{A}$ algoritmus teljesen polinomiális idejű approximációs séma *(fully polynomial-time approximation scheme, FPTAS)*

$\mathcal{P}$-re, hogyha $\mathcal{P}$ minden x esetére és minden $r > 1$ racionális számra az $\mathcal{A}$ az (x, r) inputra megadja az x egy r-approximációs megoldását $|x|$ és $1/(r-1)$ szerint is polinomiális időben.

3.1.32. Definíció. FPTAS osztály.
Az FPTAS *azon NPO-beli problémák osztálya, amelyekre van teljesen polinomiális idejű approximációs séma.*

Egy NP-nehéz optimalizálási probléma esetén egy FPTAS létezése azt bizonyítja, hogy a megoldás nehézségének ellenére a megoldást a legtöbb gyakorlati alkalmazás szempontjából hatékonyan és tetszőlegesen közelíthetjük.

A definiált approximációs osztályokra a következő tartalmazások teljesülnek:

$$FPTAS \subseteq PTAS \subseteq APX \subseteq NPO,$$

és amennyiben P $\neq$ NP, a tartalmazások szigorúak. (Van Leeuwen és Van Leeuwen, 2012)

Erősen NP-nehéz optimalizálási probléma

Bármely NPO-beli $\mathcal{P}$ probléma bármely x esetére jelölje $\max(x)$ az x-ben előforduló legnagyobb szám értékét. Vegyük észre, hogy formálisan a max függvény definíciója függ az eset kódolásától.

Legyen $\mathcal{P}$ egy NPO-beli probléma, és legyen p egy polinom. Jelölje $\mathcal{P}^{\max,p}$ azt a problémát, amelyet $\mathcal{P}$-ből kapunk úgy, hogy az eseteket azokra az x-ekre korlátozzuk, amelyekre $\max(x) \leq p(|x|)$. A következő definíció formálisan megadja azokat az optimalizálási problémákat, amelyek számítási nehézsége nem függ az eseteiben lévő számok értékétől.

3.1.33. Definíció. Erősen NP-nehéz.
Azt mondjuk, hogy egy NPO-beli $\mathcal{P}$ probléma erősen NP-nehéz *(strongly NP-hard), hogyha létezik olyan p polinom, amelyre $\mathcal{P}^{\max,p}$ NP-nehéz.*

A következő tétel egy általános feltételt ad meg, amely segít eldönteni, hogy egy problémára létezik-e FPTAS.

3.1.34. Tétel. *Legyen $\mathcal{P}$ egy erősen NP-nehéz probléma, amelyre van egy p polinom, hogy minden x inputra $m^*(x) \leq p(|x|, \max(x))$. Ha $P \neq NP$, akkor $\mathcal{P}$ nem tartozik az FPTAS osztályba.*

Optimalizálási problémák teljessége

Habár a legtöbb NP-nehéz optimalizálási problémához tartozó eldöntési probléma polinomiális időben Karp-redukálható egymásra, az optimalizálási problémák nem rendelkeznek ugyanazokkal az approximálhatósági tulajdonságokkal. Ennek elsősorban az az oka, hogy a Karp-redukciók nem minden esetben őrzik meg a mértékfüggvényt, és ha ez még meg is történik, csak ritkán őrzik meg a megoldás minőségét.

Emiatt egy erősebb redukálhatóság fogalomra van szükségünk, amely nem csupán hozzárendeli a $\mathcal{P}_1$ probléma eseteit egy $\mathcal{P}_2$ probléma eseteihez, hanem visszafelé is a $\mathcal{P}_2$ jó megoldásait $\mathcal{P}_1$ jó megoldásaihoz rendeli hozzá. Erre bevezetjük az *approximációt megőrző* (approximation preserving, AP) redukálhatóságot, amely segít két optimalizálási probléma approxmiálhatósági tulajdonságainak az összehasonlításában, és amelynek segítsgével formálisan megfogalmazhatjuk, hogy egy optimalizálási problémát nem nehezebb közelíteni, mint egy másikat.

3.1.35. Definíció. AP-redukció.
Legyen $\mathcal{P}_1$ és $\mathcal{P}_2$ kettő NPO-beli optimalizálási probléma. Azt mondjuk, hogy $\mathcal{P}_1$ AP-redukálható *(AP-reducible) $\mathcal{P}_2$-re, jelölésben $\mathcal{P}_1 \leq_{AP} \mathcal{P}_2$, ha létezik f és g két függvény, valamint egy $\alpha \geq 1$ pozitív konstans, amelyre:*

1. *Bármely $x \in I_{\mathcal{P}_1}$ esetre és bármely $r > 1$ racionális számra $f(x, r) \in I_{\mathcal{P}_2}$.*
2. *Bármely $x \in I_{\mathcal{P}_1}$ esetre és bármely $r > 1$ racionális számra, ha $SOL_{\mathcal{P}_1}(x) \neq \emptyset$, akkor $SOL_{\mathcal{P}_2}(f(x, r)) \neq \emptyset$.*
3. *Bármely $x \in I_{\mathcal{P}_1}$ esetre, bármely $r > 1$ racionális számra és bármely $y \in SOL_{\mathcal{P}_2}(f(x, r))$ megengedett megoldásra $g(x, y, r) \in SOL_{\mathcal{P}_1}(x)$.*
4. *f és g az $\mathcal{A}_f$ és $\mathcal{A}_g$ algoritmusok által polinomiális időben számolható minden rögzített r racionális számra.*
5. *Bármely $x \in I_{\mathcal{P}_1}$ esetre, bármely $r > 1$ racionális számra és bármely $y \in SOL_{\mathcal{P}_2}(f(x, r))$ megengedett megoldásra*

$$R_{\mathcal{P}_2}(f(x, r), y) \leq r \;\Rightarrow\; R_{\mathcal{P}_1}(x, g(x, y, r)) \leq 1 + \alpha(r - 1).$$

Utóbbi feltételt nevezzük AP-feltételnek.

Azt mondjuk, hogy az (f, g, α) hármas egy AP-redukció *(AP-reduction) $\mathcal{P}_1$-ről $\mathcal{P}_2$-re.*

Az approximálhatóság megőrzését az alábbi eredmény támasztja alá.

3.1.36. Lemma. *Ha $\mathcal{P}_1 \leq_{AP} \mathcal{P}_2$ és $\mathcal{P}_2 \in APX$ ($\mathcal{P}_2 \in PTAS$), akkor $\mathcal{P}_1 \in APX$ ($\mathcal{P}_1 \in PTAS$).*

A következő definíció az approximáció megőrző redukálhatóság egy eltérő megközelítését adja meg. Habár nincs olyan erős, mint az AP-redukálhatóság, a gyakorlatban mégis nagyon hasznos az AP-redukciók létezésének bizonyítására. A fogalom a lineáris kapcsolaton alapszik mind az optimális mértékek, mind az abszolút hibák között, innen jön az L-redukálhatóság elnevezés is. Mi a következő fejezetben a MAX SNP-teljesség definíciójánál alkalmazzuk.

3.1.37. Definíció. L-redukció.
Legyen $\mathcal{P}_1$ és $\mathcal{P}_2$ két NPO-beli probléma. Azt mondjuk, hogy $\mathcal{P}_1$ L-redukálható *(L-reducible) $\mathcal{P}_2$-re, jelölésben $\mathcal{P}_1 \leq_L \mathcal{P}_2$, hogyha létezik f és g függvény, valamint α és β konstans, amelyekre:*

1. *Bármely $x \in I_{\mathcal{P}_1}$ esetre $f(x) \in I_{\mathcal{P}_2}$ polinomiális időben számolható.*
2. *Bármely $x \in I_{\mathcal{P}_1}$ esetre, ha $SOL_{\mathcal{P}_1}(x) \neq \emptyset$, akkor $SOL_{\mathcal{P}_2}(f(x)) \neq \emptyset$.*
3. *Bármely $x \in I_{\mathcal{P}_1}$ esetre és $y \in SOL_{\mathcal{P}_2}(f(x))$ megengedett megoldásra $g(x,y) \in SOL_{\mathcal{P}_1}(x)$ polinomiális időben számolható.*
4. *Bármely $x \in I_{\mathcal{P}_1}$ esetre $m^*_{\mathcal{P}_2}(f(x)) \leq \beta m^*_{\mathcal{P}_1}(x)$.*
5. *Bármely $x \in I_{\mathcal{P}_1}$ esetre és $y \in SOL_{\mathcal{P}_2}(f(x))$ megengedett megoldásra*

$$\left|m^*_{\mathcal{P}_1} - m_{\mathcal{P}_1}(x, g(x,y))\right| \leq \gamma \left|m^*_{\mathcal{P}_2} - m_{\mathcal{P}_2}(f(x), y))\right|$$

Azt mondjuk, hogy az (f, g, β, γ) négyes egy L-redukció *(L-reduction) $\mathcal{P}_1$-ről $\mathcal{P}_2$-re.*

Az AP-redukálhatóság tranzitív, így egy parciális rendezést ad meg az azonos osztályban lévő problémák között. A teljes probléma fogalma megfeleltethető egy maximális elemnek a rendezésnek megfelelően. Egyúttal, ha belátjuk egy probléma teljességét, akkor egyértelmű válik, hogy felesleges jobb approximációs algoritmusok után kutatnunk.

3.1.38. Definíció. $\mathcal{C}$-nehéz, $\mathcal{C}$-teljes optimalizálási probléma.
Legyen adott NPO-beli problémák egy $\mathcal{C}$ osztálya. Egy $\mathcal{P}$ probléma $\mathcal{C}$-nehéz *($\mathcal{C}$-hard) az AP-redukálhatóság szerint, hogyha bármely $\mathcal{P}' \in \mathcal{C}$ problémára $\mathcal{P}' \leq_{AP} \mathcal{P}$. Egy $\mathcal{C}$-nehéz probléma* $\mathcal{C}$-teljes *($\mathcal{C}$-complete) az AP-redukálhatóságra nézve, hogyha $\mathcal{C}$-beli.*

A 3.1.36. Lemma és az előző definíció értelmében, NPO-beli problémák bármely $\mathcal{C}$ osztályára, amelyre $\mathcal{C} \not\subseteq$ APX ($\mathcal{C} \not\subseteq$ PTAS), ha $\mathcal{P}$ egy $\mathcal{C}$-teljes probléma, akkor $\mathcal{P} \notin$ APX

($\mathcal{P} \notin$ PTAS). Ekképpen, a $\mathcal{C}$-teljes problémák valóban a legnehezebbek a $\mathcal{C}$ osztálon belül. Ha például egy $\mathcal{P}$ probléma NPO-teljes (APX-teljes), akkor $\mathcal{P} \notin$ APX ($\mathcal{P} \notin$ PTAS), ha P $\neq$ NP. Így a teljességi eredmények egyből nem-approximálhatósági eredményeket vonnak maguk után.

MAX SNP osztály

A MAX SNP osztályt Papadimitriou és Yannakakis (1991) vezette be. A következő definíciókhoz és a kapcsolódó tételhez a hivatkozott tanulmányt követjük. A szerzők Fagin (1974) NP-re adott szintaktikus definíciójából indulnak ki. Ennek megfelelően NP az összes olyan G struktúrán értelmezett predikátumból áll, amelyek felírhatóak $\exists S \phi(G, S)$ alakban. Itt a G input, az S a megoldás, amely egy struktúra (és így $\exists S$ másodrendű kvantor), és ϕ elsőrendű formula (predikátumról, első-, valamint másodrendű logikáról lásd Ferenczi (2014)). Ebben ϕ felírható $\forall \bar{x} \exists \bar{y} \psi(\bar{x}, \bar{y}, G, S)$ alakban, ahol ψ kvantormentes (kvantorok a $\forall$ és $\exists$ szimbólumok), $\bar{x}$ és $\bar{y}$ pedig klózok.

Például, a korábban tárgyalt SAT probléma (korlátlanul hosszú klózokkal) felírható

$$\exists T \forall c \exists x \left[(P(c, x) \wedge x \in T) \vee (N(c, x) \wedge x \notin T) \right]$$

formában, ahol $P(c, x)$ azt jelenti, hogy az x változó pozitívan jelenik meg a c klózban, míg $N(c, x)$ jelenti azt, hogy negatívan, T pedig az IGAZ logikai értékű változók halmaza.

A 3SAT probléma eggyel kevesebb kvantor használatával is felírható, ha feltesszük, hogy az input négyféle relációt tartalmazhat. Legyenek ezek C_0, C_1, C_2, C_3, ahol C_j tartalmazza az összes j darab negatív literált tartalmazó klózt, vagyis $(x_1, x_2, x_3) \in C_j$ annak felel meg, hogy van egy olyan klóz, amelyben $x_1, \ldots, x_j$ negatívan jelenik meg, $x_{j+1}, \ldots, x_3$ pedig pozitívan. Ekkor a 3SAT predikátum a következő formában adható meg[5]:

$$\exists T \forall (x_1, x_2, x_3) \big[\big((x_1, x_2, x_3) \in C_0 \to x_1 \in T \vee x_2 \in T \vee x_3 \in T \big) \wedge \ldots \\ \wedge \big((x_1, x_2, x_3) \in C_3 \to x_1 \notin T \vee x_2 \notin T \vee x_3 \notin T \big) \big].$$

Azok a problémák, amelyek kifejezhetőek $\exists S \forall \bar{x} \psi(\bar{x}, G, S)$ alakban, ahol ψ kvantormentes, mint ahogyan a 3SAT is, az NP osztályon belül egy részosztályt alkotnak amelyet *szigorú NP-nek* (strict NP, SNP) nevezünk.

[5] A kifejezésben a "$\to$" szimbólum segítségével egy logikai állítást adunk meg, melyet implikációnak nevezünk. Az "$A \to B$" logikai állítás értéke HAMIS, ha A IGAZ és B HAMIS, minden más esetben IGAZ.

3.1.39. Definíció. MAX SNP osztály.
Legyen $\mathcal{P} \in SNP$ eldöntési probléma. Jelölje $\max \mathcal{P}$ *optimalizálási probléma a $\mathcal{P}$ probléma maximalizálási verzióját, amelyet a következőképpen definiálunk:*

$$\max_S \left|\left\{\bar{x} : \psi(\bar{x}, G, S)\right\}\right|.$$

MAX SNP az összes $\max \mathcal{P}$ *maximalizálási probléma osztálya, amelyre $\mathcal{P} \in SNP$.*

Vegyük észre, hogy a $\max \mathcal{P}$ maximalizálási problémában ahelyett, hogy egy olyan S megoldáshoz ragaszkodnánk, amely kielégíti ψ-t minden $\bar{x}$ klózra, egy olyan S megoldást keresünk, amely maximalizálja azon $\bar{x}$-ek számát, amelyeket kielégít. A következő tétel az approximálhatóságra vonatkozóan ad meg egy pozitív eredményt.

3.1.40. Tétel. *Minden MAX SNP-beli probléma közelíthető egy rögzített arányú approximációs algoritmussal.*

3.1.41. Definíció. MAX SNP-teljes.
Egy $\mathcal{P} \in$ MAX SNP optimalizálási probléma MAX SNP-teljes, hogyha bármely másik $\mathcal{P}' \in$ MAX SNP problémára létezik L-redukció $\mathcal{P}'$-ről $\mathcal{P}$-re.

Arora és szerzőtársai (1998) belátja, hogy amennyiben P $\neq$ NP, úgy a MAX SNP-teljes problémákra nem létezik PTAS. (Van Leeuwen és Van Leeuwen, 2012)

3.2. Hagyományos klaszterezési problémák

Először azokat a problémákat vesszük sorra bonyolultságelméleti szempontból, amelyek esetén semmilyen korlátot nem írunk elő az egyes csoportok elemszámaira.

Drineas és szerzőtársai (2004) a hagyományos klaszterezési feladatot *diszkrét klaszterezési problémának* (discrete clustering problem, DCP) nevezik, és felírásuk alapján az optimailizálási probléma a következőképpen adható meg:

3.2.1. Optimalizálási probléma. Diszkrét klaszterezési probléma (DCP).
Input: *m darab pont egy $\mathcal{A} = \{A_{(1)}, A_{(2)}, \ldots, A_{(m)}\}$ halmaza a d-dimenziós euklideszi térben (m és d nem rögzített) és egy k pozitív egész.*
Output: *k darab klaszter középpontjait tartalmazó $\mathcal{B} = \{B_{(1)}, B_{(2)}, \ldots, B_{(k)}\}$ halmaz,*

melynek elemei szintén d-dimenziós térbeli pontok, és amely minimalizálja az

$$f_{\mathcal{A}}(\mathcal{B}) = \sum_{i=1}^{m} \operatorname{dist}^2 \left(A_{(i)}, \mathcal{B}\right)$$

függvényt, ahol $\operatorname{dist}\left(A_{(i)}, \mathcal{B}\right)$ *az* $A_{(i)}$ *pont és a* $\mathcal{B}$ *halmaz* $A_{(i)}$ *ponthoz legközelebbi elemének távolsága.*

Ebben az esetben a klaszterek implicit módon adottak, mivel valamennyi $A_{(i)}$ pontot ahhoz a klaszterhez rendeljük, amelynek középpontjához a legközelebb van. A szerzők adnak egy bizonyítást arra, hogy a DCP probléma NP-nehéz, ugyanakkor ahogy arra Dasgupta (2008) és Aloise és szerzőtársai (2009) is rámutat, a levezetésben elkövetnek egy hibát.

Dasgupta (2008) az előzővel ekvivalens módon határozza meg a klaszterezési problémát általános k-ra. Ez a feladat gyakrabban használt felírása, amelyre a szakirodalom *k-közép klaszterezésként* (k-means clustering) vagy *minimális négyzetösszegű klaszterezésként* (minimum sum-of-squared clustering, MSSC) hivatkozik.

3.2.2. Optimalizálási probléma. k-közép klaszterezés *vagy* MSSC.
Input: $x_1, \ldots, x_n \in \mathbb{R}^d$ *pontok halmaza és egy* k *egész szám.*
Output: *A pontok azon* $C_1, \ldots, C_k$ *partíciója a klaszterek* μ_j *klaszterközéppontjaival együtt, amely minimalizálja a*

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

célfüggvényt.

Belátható, hogy bármely optimális megoldás esetén μ_j megegyezik a C_j-beli pontok átlagával. Dasgupta (2008) eredménye a $k = 2$ esetre vonatkozik, amelyet három lépésben, a 3SAT és a NAE3SAT NP-teljes problémák, valamint az általa megfogalmazott Általános 2-közép (Generalized 2-means) NP-nehéz probléma felhasználásával bizonyít.

3.2.3. Tétel. *A 2-közép klaszterezés NP-nehéz optimalizálási probléma.*

Az előző tanulmánytól függetlenül egy rövidebb bizonyítással Aloise és szerzőtársai (2009) is belátja a fenti tételt, ahol a redukciót a Legsűrűbb Vágás (Densest Cut) NP-nehéz problémából végzik el.

Mahajan és szerzőtársai (2009) a síkbeli klaszterezést helyezik a középpontba. A dimenziók számát $d = 2$ értékben rögzítik, és az így kapott *síkbeli 2-közép* (planar 2-means) optimalizálási problémáról látják be, hogy NP-nehéz. A bizonyításhoz felhasználják, hogy a probléma a Síkbeli 3SAT (Planar 3SAT) NP-teljes problémából (Lichtenstein, 1982) redukálható.

Novick (2009) a klaszterezési problémák egy új osztályát vezeti be. Munkájában Fisher (1958) által megfogalmazott *csoportosítási problémát* (grouping problem) általánosítja az ℓ_p-statisztika és az ℓ_q-norma egy újszerű alklamazásával.

3.2.4. Optimalizálási probléma. Csoportosítási probléma.

Input: *$x_1, x_2, \ldots, x_n \in \mathbb{R}$ megfigyelések, $w_1, w_2, \ldots, w_n$ hozzájuk tartozó súlyok, és $m < n$ egész.*

Output: *Az $\{1, 2, \ldots, n\}$ halmaz egy $\{B_1, B_2, \ldots, B_m\}$ partíciója, amely minimalizálja a* csoportokon belüli négyzetösszegeket*, vagyis a*

$$z = \sum_{j=1}^{m} \sum_{i \in B_j} w_i \left(x_i - \bar{x}_j\right)^2$$

értéket, ahol

$$\bar{x}_j = \frac{\sum_{i \in B_j} w_i x_i}{\sum_{i \in B_j} w_i}.$$

Az $\bar{x}_j$ érték az $\{x_i : i \in B_j\}$ értékek súlyozott átlaga, és azt a $\{B_1, B_2, \ldots, B_m\}$ partíciót, amely minimalizálja z-t, *legkisebb négyzetek partíciónak* (least squares partition) nevezzük.

3.2.5. Definíció. (Súlyozott) ℓ_q-norma.

Legyen $x_1, x_2, \ldots, x_n$ egy paraméter megfigyeléseinek halmaza, és legyenek $w_1, w_2, \ldots, w_n$ hozzájuk tartozó súlyok. A súlyozott ℓ_q-norma *(weighted ℓ_q-norm) $q > 0$ esetén*

$$\left[\sum_{i=1}^{n} w_i |x_i|^q\right]^{\frac{1}{q}}.$$

A súlyozott ℓ_∞-norma *értéke* $\max_{i=1,\ldots,n} w_i x_i$.

3.2.6. Definíció. (Súlyozott) ℓ_p-statisztika.

Legyen $x_1, x_2, \ldots, x_n$ egy paraméter megfigyeléseinek halmaza, és legyenek $w_1, w_2, \ldots, w_n$ hozzájuk tartozó súlyok. $p > 0$ esetén a súlyozott ℓ_p-statisztika *(weighted ℓ_p-statistic) a*

$$\min \left[\sum_{i=1}^{n} w_i |x_i - c|^p\right]^{\frac{1}{p}}$$

célfüggvény egy c megoldása.

$p > 1$-re a célfüggvény szigorúan konvex, ennek következtében $p > 1$ esetén az ℓ_p-statisztika egyértelmű.

3.2.7. Optimalizálási probléma. $\ell_q - \ell_p$ optimalizálási probléma.

Input: *$x_1, x_2, \ldots, x_n \in \mathbb{R}$, hozzájuk tartozó $w_1, w_2, \ldots, w_n$ nemnegatív súlyok és $m < n$ egész szám.*

Output: *Az $\{1, 2, \ldots, n\}$ halmaz egy $\{B_1, B_2, \ldots, B_m\}$ partíciója, amely minimalizálja a*

$$\sum_{j=1}^{m} \sum_{i \in B_j} w_i \|x_i - c_j\|^q$$

célfüggvényt, ahol c_j a B_j-hez tartozó súlyozott ℓ_p-statisztika.

3.2.8. Tétel. *Bármely $0 < q \leq p - 1$ esetén az ℓ_q-ℓ_p optimalizálási probléma NP-nehéz.*

A bizonyításhoz a Részhalmaz-összeg (Subset-sum) NP-teljes problémát (Garey és Johnson, 1979) használja fel. Ebből az eredményből már következik az is, hogy az ℓ_q-ℓ_p optimalizálási probléma magasabb dimenzióban is NP-nehéz. Ugyanis, ha bármilyen $d > 2$ dimenzióban lenne megfogalmazva a probléma, az magában foglalja azt az esetet is, amikor a d-dimenziós vektorok első koordinátái felelnek meg az $x_1, \ldots, x_n$ értékeknek, a többi index helyén pedig 0-k állnak, amely pedig ekvivalens a megfogalmazott egydimenziós problémával.

Kel'manov és Pyatkin (2016) ugyancsak a 2-klaszterezéssel foglalkozik, és három optimalizálási problémát fogalmaznak meg. Ezek közül az első kettő, a *kvadratikus euklideszi minimális összegű párok 2-klaszterezés* (Quadratic Euclidean Min-Sum All-Pairs 2-clustering) és az *Euklideszi egyenletes variancia alapú 2-klaszterezés* (Euclidean Balanced Variance-based 2-clustering) a 2-közép klaszterezéssel ekvivalens. A szerzők mindkettőről belátják, hogy erősen NP-nehéz, amelynek bizonyításához a Kvadratikus Euklideszi Maximális Vágás (Quadratic Euclidean Max-Cut) (Ageev és szerzőtársai, 2014) erősen NP-nehéz problémát használják fel.

A harmadik optimalizálási probléma során az egyik klaszterközéppontot előre adottnak tekintjük.

3.2.9. Optimalizálási probléma. Euklideszi egyenletes variancia-alapú 2-klaszterezés egy adott középponttal (Euclidean Balanced Variance-based 2-clustering with a given

center).

Input: *$\mathbb{R}^d$-beli pontok $\mathcal{Y} = \{y_1, \ldots, y_n\}$ halmaza.*

Output: *Az $\mathcal{Y}$ halmaz partíciója két nemüres $\mathcal{X}$ és $\mathcal{Z}$ halmazra, amelyekre a*

$$f(\mathcal{X}, \mathcal{Z}) = |\mathcal{X}| \sum_{x \in \mathcal{X}} \|x - \bar{x}(\mathcal{X})\|^2 + |\mathcal{Z}| \sum_{z \in \mathcal{Z}} \|z\|^2$$

célfüggvény értéke minimális, ahol $\bar{x}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} x$ az $\mathcal{X}$ csoport geometriai középpontja.

3.2.10. Tétel. *Az euklideszi egyenletes variancia-alapú 2-klaszterezés egy adott középponttal erősen NP-nehéz.*

A szerző a bizonyításhoz felhasználja, hogy a Minimális Biszekció (Minimum Bisection) NP-nehéz probléma (Garey és szerzőtársai (1976): Minimális Vágás Egyenlő Méretű Részhalmazokra (Minimum Cut into Equal-sized Subsets) vagy általánosan Garey és Johnson (1979): Minimális Vágás Korlátos Halmazokra (Minimum Cut into Bounded Sets)) visszavezethető a fenti feladatra.

3.3. Klaszterezés elemszám megkötésekkel

A problémakörhöz kapcsolódó egyik első eredmény Edmonds (1965a) algoritmusa, amelyben polinomiális futásidejű algoritmust adott a maximális méretű párosítás megtalálására.[6] Edmonds (1965b) tanulmányában pedig leírja az első olyan eljárást is, amely polinomiális futádisőben találja meg a maximális súlyú teljes párosítást élsúlyozott gráfban, ami által egyértelmű, hogy a probléma P-beli. Mivel felírásában az élsúlyok tetszőleges valós számok, ezért ebből egyből következik a minimális súlyú teljes párosítás P-belisége is. Ugyanis az élsúlyok -1-szeresét véve a feladat visszavezethető a maximális súlyú teljes párosítás keresésére, amely polinomiális futásidőben megoldható.

A *Particionálás háromszögekre* (Partition into Triangles) NP-teljes probléma elsőként Karp (1975), valamint Garey és Johnson (1979) forrásokban jelenik meg. Utóbbi felírásában az eldöntési probléma a következő:

3.3.1. Eldöntési probléma. Particionálás háromszögekre.

Input: *$G = (V, E)$ gráf, a csúcsok száma $|V| = 3q$ valamilyen q egészre.*

[6] A későbbiekben számos jobb, szintén polinomiális futásidejű algoritmus is megjelent, melyekről például Lovász és Plummer (1986) ad áttekintést.

Kérdés: *Van-e a G gráfnak olyan q diszjunkt halmazból álló $V_1, V_2, ..., V_q$ partíciója, amely esetén bármely $V_i = \{u_i, v_i, w_i\}, 1 \leq i \leq q$ halmazra az $\{u_i, v_i\}, \{u_i, w_i\}$ és $\{v_i, w_i\}$ élek mindegyike E-beli.*

Kirkpatrick és Hell (1978, 1983) a vizsgabeosztások problémájához kapcsolódóan, általánosan tekintették a párosításokat. Felírásuk szerint legyen adott egy H tetszőleges gráf, valamint jelölje $V(H)$ és $E(H)$ rendre a H csúcsainak és éleinek halmazát. Az $E(H)$ élek egy M halmazát H-beli *párosítás*nak (matching) nevezzük, hogyha semelyik két M-beli elemnek sincs közös csúcsa. Egy M párosítás *teljes* (perfect), vagy másnéven *1-faktor* (1-factor), hogyha a $V(H)$-beli csúcsok mindegyikét az M-beli elemek közül csak pontosan egy fedi. Ha H egy súlyozott gráf, akkor az M párosítás súlyát értelemszerűen a $\sum_{e \in M}$ súly(e) összeg adja.

Egy H-beli párosításra a H algráfjainak diszjunkt halmazaként is tekinthetünk, ahol minden egyes algráf izomorf K_2-vel, vagyis a 2-csúcsú teljes gráffal. Egy teljes párosítás esetén a $V(H)$ csúcshalmazt teljesen particionálják az algráfok csúcshalmazai. Ezek alapján természetes módon adódik a következő általánosítás.

3.3.2. Definíció. G-pakolás, G-faktor.
*Legyen G egy tetszőleges gráf. Egy G-*pakolás *(G-packing) egy H gráf esetén a H diszjunkt részgráfjainak egy $\{G_1, ..., G_d\}$ halmaza, ahol minden egyes G_i gráf izomorf G-vel. Egy teljes G-pakolás vagy G-*faktor *(G-factor) egy H gráf esetén egy olyan G-pakolás, ahol a $V(G_i)$ halmazok a $V(H)$ egy partícióját adják.*

Ennek megfelelően a K_2-pakolás egy párosítás, a K_2-faktor pedig egy teljes párosítás. Ez alapján a következő, FACT(G) eldöntési problémát írják fel:

3.3.3. Eldöntési probléma. FACT(G).
Input: *Egy H gráf.*
Kérdés: *Van-e a H gráfnak G-faktora?*

Vezessük be a következő jelölést. $G = \alpha \cdot K_1 \cup \beta \cdot K_2$ akkor, hogyha a G gráf α darab K_1 és β darab K_2 diszjunkt uniójából áll, vagy másképpen, ha G bármely összefüggő komponensében legfeljebb két csúcs található. Kirkpatrick és Hell (1978, 1983) a következő tételt látja be a 3-dimenziós párosítás (3-dimensional matching, Karp (1972)) és az annak általánosításaként adódó k-dimenziós párosítás NP-teljes problémák segítségével.

3.3.4. Tétel. *Ha G nem $\alpha \cdot K_1 \cup \beta \cdot K_2$ alakú, akkor a FACT(G) probléma NP-teljes.*

A fenti megfogalmazás olyan eseteket is megenged, amikor egy G-faktor egy G_i eleme nem feszített részgráfja a H gráfnak, tehát van a H-nak olyan éle, amely G_i-beli csúcsok között húzódik, viszont nem része G_i-nek. A szigorúbb esetet Kirkpatrick és Hell (1978, 1983) az alábbiak szerint tárgyalja.

3.3.5. Definíció. Szigorú G-pakolás (vagy G-faktor).
Egy H gráfra vonatkozó G-pakolás (vagy G-faktor) szigorú *(strict), ha a pakoláshoz tartozó valamennyi G_i gráf a H gráf egy feszített részgráfja.*

3.3.6. Eldöntési probléma. S-FACT(G).
Input: *Egy H gráf.*
Kérdés: *Van-e a H gráfnak szigorú G-faktora?*

3.3.7. Tétel. *Ha G-nek legalább 3 csúcsa van, akkor az S-FACT(G) probléma NP-teljes.*

A G-pakolás fogalmát kiterjeszthetjük, ha a G gráfot lecseréljük lehetséges gráfok egy $\mathcal{G}$ halmazára.

3.3.8. Definíció. $\mathcal{G}$-pakolás, $\mathcal{G}$-faktor.
Egy H gráf egy $\mathcal{G}$-pakolása *egy olyan H diszjunkt részgráfjaiból álló $\{G_1, \ldots, G_d\}$ halmaz, amely esetén minden G_i izomorf $\mathcal{G}$ valamely elemével. A* $\mathcal{G}$-faktor *hasonlóan definiálható.*

A $\mathcal{G}$-faktorra vonatkozó eldöntési probléma értelemszerűen következik.

3.3.9. Eldöntési probléma. FACT($\mathcal{G}$).
Input: *Egy H gráf.*
Kérdés: *Van-e a H gráfnak $\mathcal{G}$-faktora?*

3.3.10. Tétel. *Legyen $\mathcal{G}$ a $\{K_t | t \geq 1\}$ halmaz valamely részhalmaza. Ha $K_1 \in \mathcal{G}$ vagy $K_2 \in \mathcal{G}$, akkor FACT($\mathcal{G}$) P-beli, egyébként pedig FACT($\mathcal{G}$) NP-teljes.*

Ennek megfelelően, ha legalább három férőhelyes csoportok állnak rendelkezésre, akkor az elemek beosztásának megfelelő FACT($\{K_t | t \geq 3\}$) probléma NP-teljes.

Kann (1991) megmutatja, hogy egy gráf maximális fedése 3-elemű csúcshalmazokkal MAX SNP-nehéz, hogyha pedig a csúcsok fokszáma felülről korlátos egy konstanssal,

akkor pedig MAX SNP-teljes is. Ez a tanulmány első eredményéből következik, mely szerint az NP-teljes 3-dimenziós párosítási probléma optimalizálási verziója MAX SNP-nehéz. Ezt a maximális korlátos 3SAT (MAX 3SAT-B) problémából történő redukcióval látja be, amelyről ismert, hogy MAX SNP-teljes (Papadimitriou és Yannakakis, 1991).

3.3.11. Optimalizálási probléma. Maximális fedés háromszögekkel - B.
Input: *Egy $G = (V, E)$ gráf, ahol a csúcsok fokszáma felülről korlátos B-vel.*
Output: *G legnagyobb fedése háromszögekkel, vagyis azon legnagyobb méretű $\{V_i\}$ halmazt, amelynek elemei diszjunkt háromelemű csúcshalmazok úgy, hogy valamennyi V_i egy 3-klikk (3-clique, K_3) G-ben.*

3.3.12. Következmény. A Maximális fedés háromszögekkel - B probléma $B \geq 6$ esetén MAX SNP-teljes.

A maximális fedés háromszögekkel elnevezés helyett gyakran találkozunk a *Maximális háromszög pakolás* (Maximum Triangle Packing, MTP) névvel, amely ugyanarra az optimalizálási problémára utal. Szintén ezt tekinti Caprara és Rizzi (2002) két különböző felírásban.

3.3.13. Definíció. Háromszög csúcs-pakolás, háromszög él-pakolás.
Egy $G = (V, E)$ gráfban a G háromszögeinek egy $T_1, \ldots, T_k$ családját háromszög csúcs-pakolásnak *(node-packing of triangles) nevezzük, hogyha a $T_1, \ldots, T_k$ háromszögek csúcs-diszjunktak. Hasonlóan,* háromszög él-pakolásnak *(edge-packing of triangles) nevezzük, hogyha a $T_1, \ldots, T_k$ háromszögek él-diszjunktak.*

A definícióknak megfelelően két optimalizálási problémát vizsgálnak, amelyek közül az első a Kirkpatrick és Hell (1978, 1983) által megfogalmazott FACT(G) problémával állítható párhuzamba $G = K_3$ gráfra. Ugyanakkor, míg korábban egy G-faktor, vagyis az eredeti gráf összes csúcsát fedő pakolás létezésére vonatkozó eldöntési problémával álltunk szemben, jelen esetben a lefedhető csúcsok maximális száma a kérdés.

3.3.14. Optimalizálási probléma. Csúcs-diszjunkt háromszög pakolás *(Node-disjoint Triangle Packing, NTP).*
Input: *Egy G gráf.*
Output: *Maximális méretű háromszög csúcs-pakolás.*

3.3.15. Optimalizálási probléma. Él-diszjunkt háromszög pakolás (Edge-disjoint Triangle Packing, ETP).

Input: *Egy G gráf.*

Output: *Maximális méretű háromszög él-pakolás.*

Hurkens és Schrijver (1989) egy általános eredményéből következik, hogy mind az NTP, mind az ETP problémára bármely $\epsilon > 0$ esetén létezik polinomiális futásidejű $(3/2 + \epsilon)$-approximációs algoritmus (Caprara és Rizzi, 2002), amelyből következik, hogy az optimalizálási problémák APX-beliek.

Jelölje $\Delta(G)$ a G gráf maximális fokszámát, vagyis $\Delta(G) = \max_{v \in V} |\delta(v)|$, ahol $\delta(v)$ jelöli azon élek halmazát, amelyeknek egyik végpontja a $v \in V$ csúcs. Nevezzünk egy gráfot *irredundánsnak* (irredundant), hogyha bármely éle része egy háromszögnek. Caprara és Rizzi (2002) a következő pozitív állításokat fogalmazza meg irredundáns G gráfokra.

3.3.16. Állítás. *Az NTP optimalizálási probléma polinomiálisan megoldható $\Delta(G) \leq 3$ esetén.*

3.3.17. Állítás. *Az ETP optimalizálási probléma polinomiálisan megoldható $\Delta(G) \leq 4$ esetén.*

Ezután a szerzők a nehézségi eredményeket tárgyalják. Az az állítás, miszerint az NTP feladat APX-nehéz már $\Delta(G) = 4$ értékre is, már korábban bizonyítást nyert. Az alábbi eredmények bizonyításához az APX-nehéz MAX-2-SAT-3 problémából (Ausiello és szerzőtársai, 2003; Berman és Karpinski, 1999) történő redukciót alkalmaznak.

3.3.18. Állítás. *Az ETP optimalizálási probléma APX-nehéz már $\Delta(G) = 5$ esetén is.*

Mivel az ETP és NTP problémák APX-beliek, ezért az APX-nehézségből az APX-teljesség is következik. További negatív eredményeknek tekinthetőek a *síkbarajzolható* (planar) gráfokra vonatkozó állítások. Egy gráfot síkbarajzolhatónak nevezünk, ha le lehet rajzolni úgy a síkban, hogy az élek nem metszik egymást, azaz csak a csúcsoknál találkoznak. A bizonyításokhoz az NP-teljes Síkbeli 3SAT problémát (Lichtenstein, 1982) használják fel.

3.3.19. Állítás. *Az NTP optimalizálási probléma NP-nehéz síkbarajzólható gráfokra már $\Delta(G) = 4$ esetén is.*

3.3.20. Állítás. *Az ETP optimalizálási probléma NP-nehéz síkbarajzolható gráfokra már* $\Delta(G) = 5$ *esetén is.*

Chataigner és szerzőtársai (2009) szintén egy háromszög pakolási feladatot tekint. Felírásukban az *$\mathcal{F}$-pakolási probléma* a maximális számú él megtalálása, amely egy $\mathcal{F}$-pakolással lefedhető. Legyen $\mathcal{K}_r = \{K_2, \ldots, K_r\}$. A szerzők a legfeljebb 4-es fokszámmal rendelkező, irredundáns gráfokra vonatkozó APX-teljes $\{K_3\}$-pakolás problémát (lásd Caprara és Rizzi (2002) előzőleg tárgyalt eredményeinél az NTP problémát) felhasználva belátják a következőeket.

3.3.21. Tétel. *A $\mathcal{K}_3$-pakolás probléma APX-nehéz olyan gráfokon, ahol a csúcsok fokszámának maximuma 5.*

3.3.22. Tétel. *A $\mathcal{K}_3$-pakolás probléma APX-nehéz azon irredundáns gráfok osztályán, ahol a csúcsok fokszámának maximuma 4.*

Az élsúlyozott gráfokra vonatkozó első általános eredményt Feo és Khellaf (1990) bizonyította. Tanulmányukban megfogalmazzák az m-dimenziós párosítás gráfon (mDM) problémát, amely az élsúlyozott optimalizálási problémához tartozó eldöntési probléma. Erről a Gráf Particionálás (Graph Partitioning) NP-teljes problémából (Garey és Johnson, 1979) történő redukcióval belátják, hogy NP-teljes.

3.3.23. Eldöntési probléma. m-dimenziós párosítás gráfon (mDM).
Input: *Egy $G = (V, E)$ gráf, ahol $|V| = km, m \geq 3$ és E minden éléhez tartozik egy w_e súly. Továbbá egy j egész szám.*
Kérdés: *Van-e olyan $V_1, V_2, \ldots, V_k$ diszjunkt, m csúcsot tartalmazó halmazokból álló partíciója a V gráfnak, hogy azon élek súlyának összege, amelyeknek mindkét végpontja ugyanabban a V_i halmazban található, nagyobb mint j?*

3.3.24. Lemma. *Az mDM probléma NP-teljes.*

Ebből pedig már következik, hogy a *maximális súlyú m-dimenziós párosítás* optimalizálási probléma NP-nehéz. Egy szigorúbb esetet fogalmaz meg az a probléma, amikor a gráf csúcsai között lévő élekre teljesül a háromszög-egyenlőtlenség. Legyen A egy $n \times n$-es, valós, szimmetrikus mátrix, amelynek főátlójában 0-k állnak, vagyis $a_{ii} = 0\ \forall i$.

Azt mondjuk, hogy A-ra teljesül a háromszög-egyenlőtlenség, ha $a_{ij} \leq a_{ik} + a_{kj}$ minden $1 \leq i, j, k \leq n$ esetén. Definiáljuk a ΔmDM eldöntési problémát úgy, mint egy olyan m-dimenziós párosítás gráfon probléma, ahol a gráf élsúlyainak mátrixára teljesül a háromszög-egyenlőtlenség.

3.3.25. Lemma. *A ΔmDM probléma NP-teljes.*

Bertoni és szerzőtársai (2012) a k-közép klaszterezés (lásd 3.2.2. Optimalizálási probléma), vagy ahogyan a tárgyalásukban röviden szerepel, a *k-klaszterezés* problémáját veszik alapul, és ezt egészítik ki elemszám megkötésekkel. A feladatot általánosan írják fel, majd egydimenziós esetekre fogalmaznak meg eredményeket. A csoportokon belüli távolságokat a p-norma, jelölésben $\|\cdot\|_p$, rögzített $p \geq 1$ értékkel határozza meg, ahol $\|(\alpha_1, \ldots, \alpha_d)\|_p = (\sum |\alpha_i|^p)^{\frac{1}{p}}$. A klaszterezési feladat eredményeképpen kapott $\{A_1, A_2, \ldots, A_k\}$ partíciót *k-klaszterezésnek* nevezik, egy A klaszter *p-középpontját* (p-centorid) pedig a következőképpen definiálják:

$$C_A = \arg\min_{\mu \in \mathbb{R}^d} \sum_{x \in A} \|x - \mu\|_p^p .$$

Ha $p > 1$, akkor a középpont egyértelmű, ha pedig $p = 2$, akkor a középpont a $C_A = (\sum_{x \in A} x)/|A|$ átlaggal egyezik meg. Ha $p = 1$, akkor több középpont is adódhat, amelyek közül az egyik a komponensenkénti medián. Egy A csoport $W(A)$ költségét a

$$W(A) = \sum_{x \in A} \|x - C_A\|_p^p$$

függvény adja meg, a csoportok méretére pedig elemszám megkötéseket írnak elő, így végül a következő optimalizálási probléma adódik:

3.3.26. Optimalizálási probléma. Elemszám Megkötéses Klaszterezési Probléma *(Size Constrained Clustering Problem, SCC).*

Input: *Legyen egy $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ ponthalmaz, egy $k > 1$ egész és $m_1, m_2, \ldots, m_k$ k pozitív egész, amelyekre $\sum_{i=1}^k m_i = n$.*

Output: *Egy $\{A_1, A_2, \ldots, A_k\}$ k-klaszterezés, amely elemeire*

$$|A_i| = m_i \ \forall i = 1, \ldots, k$$

teljesül, és amely minimalizálja a

$$W(A_1, A_2, \ldots, A_k) = \sum_{i=1}^{k} W(A_i)$$

költséget.

Ha egy SCC probléma esetén a dimenziók d számát rögzítettnek tekintjük, akkor SCC-d problémának nevezzük. Ha a csoportok k száma rögzített, akkor k-SCC problémaként hivatkozunk rá. Végül, ha d és k is rögzített, akkor a feladatot k-SCC-d problémának nevezzük.

Bertoni és szerzőtársai (2012) az SCC problémák kapcsán három komplexitásra vonatkozó eredményt közöl. A 2-SCC probléma NP-nehéz, amelynek bizonyításához a Minimum Biszekció (Minimum Bisection) NP-nehéz problémát (Garey és szerzőtársai, 1976) használják fel. A 2-SCC-1 probléma polinomiális időben megoldható, hogyha $p \geq 1$ egész. Végül pedig, az SCC-1 probléma NP-nehéz minden $p \geq 1$ esetén, amelynek bizonyításánál a *3-Particionálás* (3-Partition) NP-teljes problémából (Hulett és szerzőtársai (2008) egy ezzel ekvivalens problémára látták be az NP-teljességet) történő redukciót alkalmaznak.

Lin és szerzőtársai (2016) kiterjeszti a 2-SCC-1 problémára vonatkozó nehézségi eredményt a síkra, és megmutatja, hogy az euklideszi norma szerint tekintett célfüggvény esetében a 2-SCC-2 problémára létezik polinomiális futásidejű algoritmus az optimum megadására.

Kel'manov és Pyatkin (2016) a 2-klaszterezés optimalizálási problémára adott három változatát az előző fejezetben közöltük. Az elemszám megkötésekre vonatkozó eredményüket az alábbi következmény mondja ki.

3.3.27. Következmény. Mindhárom tárgyalt optimalizálási probléma erősen NP-nehéz akkor is, hogyha a kívánt részhalmazok számossága is az input részét képezi.

Ez abból következik, hogy az ismeretlen elemszámokra megfogalmazott optimalizálási probléma úgy is megközelíthető, hogy megoldjuk azt az $\mathcal{O}(n)$ számú feladatot, ahol a két részhalmazra valamilyen meghatározott elemszámot is előírunk.

Pyatkin és szerzőtársai (2017) az *Egyenletes MSSC* (Balanced MSSC) problémát vizsgálja, amely az MSSC problémának (lásd 3.2.2. Optimalizálási probléma) az a speciális esete, amelyben minden klaszter $m = n/k$ elemszáma megegyezik.

Az egyenletes MSSC probléma $m = 2$ esetén visszavezethető a minimális súlyú teljes párosítás keresésére, ezért Edmonds (1965b) eredménye nyomán polinomiális futásidőben megoldható. Emellett az előzőleg tárgyalt eredmények alapján azt is tudjuk, hogy két egyenlő méretű csoport, vagyis $k = 2$ esetén a probléma NP-nehéz. Pyatkin és szerzőtársai

(2017) arra a kérdésre keresi a választ, hogy az n/k hányados mely értékétől válik a probléma NP-nehézzé. Tanulmányukban belátják, hogy az optimalizálási probléma NP-nehéz, amikor a csoportonkénti elemek száma (n/k) három.

3.3.28. Tétel. *Az Egyenletes MSSC probléma NP-teljes $n/k = 3$ esetén.*

A bizonyításhoz a redukció a Particionálás háromszögekre NP-teljes problémából (Garey és Johnson, 1979) történik.

3.3.1. m-dimenziós párosítási problémák

A következőekben kiterjesztjük az egyenlő méretű csoportok létrehozására vonatkozó általános dimenziós nehézségi eredményeket. Megmutatjuk expliciten, hogy ha a csoportok m mérete legalább 3, akkor az egyenletes MSSC feladat különböző távolságfelírások mellett, a célfüggvény minimalizálására és maximalizálására is NP-nehéz. Ezzel egyúttal alternatív bizonyítást adunk a Feo és Khellaf (1990) által tárgyalt mDM problémához tartozó optimalizálási probléma NP-nehézségére is. Az optimalizálási feladatra az alábbiakban *maximális súlyú m-dimenziós párosítás* (MAX-mDM) problémaként hivatkozunk, és $m \geq 2$ egész érték mellett tekintjük. Továbbá belátjuk, hogy az előző minimalizálási verziója, a *minimális súlyú m-dimenziós párosítás* (MIN-mDM) szintén NP-nehéz, amennyiben $m \geq 3$. Utóbbi jelentősége abban rejlik, hogy az az egyenletes MSSC probléma általánosításaként is tekinthető, amelyben a pontok közötti távolságok tetszőlegesek lehetnek. A MAX-mDM és MIN-mDM problémákra összefoglalóan *m-dimenziós párosítási problémákként* hivatkozunk. Elméleti eredményeinket Kondor (2022a) műhelytanulmányban közöltük, az m-dimenziós párosítási problémákra vonatkozó összefoglaló táblázatokat pedig Kondor (2022b) tanulmányban szerepeltetjük.

A nehézségi eredmények kiterjesztéséhez az egyenletes MSSC problémából indulunk ki. Huygen tétele[7] alapján a klaszterekben belüli pontoknak a klaszter centroidjától vett négyzetes távolságainak összege megegyezik a klaszter pontjainak egymástól vett négyzetes távolságainak az adott klaszter elemszámának kétszeresével leosztott összegével. Ugyanez egyszerűbben, formálisan megadva a $C_1, \ldots, C_k$ klaszterek egységes m elemszáma esetén

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \left\| x_i - \left(\frac{1}{m} \sum\nolimits_{x_j \in C_s} x_j \right) \right\|^2 = \frac{1}{2m} \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} \|x_i - x_j\|^2.$$

[7]Huygen tételét először Edwards és Cavalli-Sforza (1965) bizonyította (Novick, 2009).

Ezt felhasználva átírhatjuk az egyenletes MSSC problémát úgy, hogy annak célfüggvényében a pontok közötti euklideszi távolságok szerepeljenek. A probléma általánosításához ezt követően lecseréljük az euklideszi távolságot egy, az ℓ_p normán alapuló távolságmértékre, valamint a feladat minimalizálási verziója mellett a maximalizálási célt is tekintjük. Az így kapott feladatokat a 3.3.29. Optimalizálási problémával definiáljuk.

3.3.29. Optimalizálási probléma. pMIN-mDM (pMAX-mDM).
Input: *$C = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ pontok halmaza, valamint m és p egész számok, ahol $k = n/m$ egész.*
Output: *A pontok azon $C_1, \ldots, C_k$ egyenlő méretű csoportokból álló partíciója, amely minimalizálja (maximalizálja) a*

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \sum_{x_j \in C_s} \|x_i - x_j\|_p^p$$

célfüggvényt.

Vegyük észre, hogy az egyenletes MSSC probléma ekvivalens a 2MIN-mDM problémával. A következőekben belátjuk a pMIN-mDM és pMAX-mDM problémák NP-nehézségét különböző p értékek mellett.

3.3.30. Tétel. (Kondor (2022a)) *A pMIN-mDM optimalizálási probléma NP-nehéz bármely rögzített $m \geq 3$ és $p \geq 1$ egészekre.*

Bizonyítás. A tétel bizonyításához belátjuk, hogy a pMIN-mDM problémához tartozó 3.3.31. Eldöntési probléma NP-teljes.

3.3.31. Eldöntési probléma. A pMIN-mDM problémához tartozó eldöntési probléma.
Input: *$C = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ pontok halmaza, valamint m és p egész számok, ahol $k = n/m$ egész, és $W \in \mathbb{Q}^+$.*
Kérdés: *Van-e a pontok C halmazának olyan $C_1, \ldots, C_k$ diszjunkt, m csúcsot tartalmazó halmazokból álló partíciója, amelyre*

$$\sum_{s=1}^{k} \sum_{x_i \in C_s} \sum_{x_j \in C_s} \|x_i - x_j\|_p^p \leq W$$

teljesül?

A probléma NP-beli. A bizonyítás hátralévő részében Pyatkin és szerzőtársai (2017) lépéseit követjük. A redukciót az m-dimenziós párosítás élsúlyozatlan gráfon (mDM-$\{0,1\}$)

eldöntési problémából végezzük el, amelyet a 3.3.32. Eldöntési probléma ír le. Erről Feo és Khellaf (1990) belátja, hogy NP-teljes az mDM probléma NP-nehézségi bizonyításának egy lépéseként.

3.3.32. Eldöntési probléma. m-dimenziós párosítás élsúlyozatlan gráfon (mDM-$\{0,1\}$).
Input: *Egy $G=(V,E)$ gráf, ahol $|V|=km, m\geq 3$, k,l pozitív egészek.*
Kérdés: *Van-e a csúcsok V halmazának olyan $V_1,V_2,\ldots,V_k$ diszjunkt, m csúcsot tartalmazó halmazokból álló partíciója, amelyre azon élek száma, amelyeknek mindkét végpontja ugyanabban a V_i halmazban található, nagyobb mint l?*

A bizonyításhoz megmutatjuk, hogy az mDM-$\{0,1\}$ probléma tetszőleges esetét meg tudnánk oldani (polinomiális időben), hogyha a pMIN-mDM problémához tartozó eldöntési problémát a redukció során kapott paraméterekkel meg tudnánk oldalni (polinomiális időben).

Tekintsük az mDM-$\{0,1\}$ probléma egy tetszőleges esetét $|V|=km$ darab csúccsal és $|E|=q$ darab éllel. Rögzítsük a p pozitív egész értékét, és legyen $d=q$, valamint $W=4q(m-1)-4(l+1)$. Jelölje $C\in\{0,1\}^{km\times d}$ a G gráf incidenciamátrixát, vagyis $x_{it}=1$, ha a $v_i\in V$ csúcs az e_t él egyik végpontja, és $x_{it}=0$ egyébként. Ekkor a C soraira tekinthetünk $\mathbb{R}^d$-beli pontokként. Ezen pontok $n/k=m$ méretű klaszterekbe történő rendezései pontosan megfeleltethetőek a gráf csúcsainak m méretű részhalmazokra való particionálásainak.

Legyen $A_{st}=\sum_{i:x_i\in C_s}\sum_{j:x_j\in C_s}|x_{it}-x_{jt}|^p$ a C_s részhalmaz t koordináta szerinti hozzájárulása a célfüggvényhez. A pMIN-mDM célfüggvényét ekkor felírhatjuk mint

$$\sum_{t=1}^{d}\sum_{s=1}^{k}A_{st}.$$

Ha két olyan x_i és x_j pontot tekintünk, amelyek t-edik koordinátája 1, akkor a következő két eset egyike teljesül rájuk: a pontok 1) ugyanabba a C_{s_1} részhalmazba tartoznak, vagy 2) különböző, C_{s_1} és C_{s_2} csoportokba lettek particionálva. Jelöle $A_t=\sum_{s=1}^{k}A_{st}$ a t-edik koordináta hozzájárulását a célfüggvényhez, és jelölje $A_t^{(1)}$, valamint $A_t^{(2)}$ ennek értékeit a két különböző esetben. Ekkor a hozzájárulás az első esetben

$$A_t^{(1)}=A_{s_1t}=2|1-1|^p+4(m-2)|1-0|^p=4(m-2),$$

míg a második esetben

$$A_t^{(2)} = A_{s_1t} + A_{s_2t} = 4(m-1)|1|^p = 4(m-1).$$

Végül jelölje b azon élek számát, amelyeknek mindkét végpontja ugyanabban a részhalmazban van. Ekkor egy egyenletes partíció célfüggvényértéke kifejezhető a b függvényeként, vagyis

$$A(b) = (q-b)A_t^{(2)} + bA_t^{(1)} = 4q(m-1) - 4b. \tag{3.2}$$

$A(b)$ csökkenő b-ben, amelyből következik, hogy $A(b) \leq W$ akkor és csak akkor, hogyha $b \geq l+1$. □

Az előző párjaként a 3.3.33. Tétel a pMAX-mDM NP-nehézségét mondja ki.

3.3.33. Tétel. (Kondor (2022a)) *A pMAX-mDM optimalizálási probléma NP-nehéz bármely rögzített $m \geq 3$ és $p \geq 2$ egészekre.*

Bizonyítás. A bizonyítás során a 3.3.30. Tétel bizonyításának lépéseit követjük a következő változtatásokkal. A pMAX-mDM optimalizálási problémához tartozó eldöntési problémában a célfüggvény értékének nagyobb, vagy egyenlőnek kell lennie, mint W.

A p értékét úgy kell megválasztani, hogy a $p \geq 2$ egyenlőtlenség is teljesüljön rá, és a célfüggvény célértéke legyen $W = 4q(m-1) + (l+1)(2^{p+1} - 4)$. Amikor a G gráf incidenciamátrixát tekintjük, akkor ebben az esetben egy módosított mátrixot adunk meg. A C minden egyes oszlopában az egyik egyes értéket cseréljük le -1-re, és hagyjuk a másik értéket változatlanul. Az előzőekből következően

$$\begin{aligned} A_t^{(1)} &= 2^{p+1} + 4(m-2), \\ A_t^{(2)} &= 4(m-1), \text{ és} \\ A(b) &= 4q(m-1) + b(2^{p+1} - 4). \end{aligned}$$

Mivel $p \geq 2$, $A(b)$ növekvő b-ben, így $A(b) \geq W$ akkor és csak akkor, hogyha $b \geq l+1$. □

Mivel az egyenletes MSSC probléma ekvivalens a 2MIN-mDM problémával, ezért a 3.3.30. Tételből egyből következik az egyenletes MSSC probléma NP-nehézsége is általános m-re.

3.3.34. Következmény. (Kondor (2022a)) Az egyenletes MSSC probléma bármely rögzített $m \geq 3$ egész esetén NP-nehéz.

A 3.3.30. Tételből szintén következik, hogy az m-dimenziós párosítási probléma minimalizálási verziója is NP-nehéz, hiszen az tartalmazza a 2MIN-mDM problémát.

3.3.35. Következmény. (Kondor (2022a)) A minimális súlyú m-dimenziós párosítás optimalizálási probléma (MIN-mDM) NP-nehéz bármely rögzített $m \geq 3$ esetén.

Az m-dimenziós párosításokra vonatkozó nehézségi eredményeket a 3.2. és 3.3. táblázatok összegzik. A kapcsolódó eredményeket zárójelben hivatkozzuk, a 3.3. táblázatban a következő számozott rövidítéseket alkalmazzuk: (1) Bertoni és szerzőtársai (2012), (2) Lin és szerzőtársai (2016), (3) Kel'manov és Pyatkin (2016), (4) Pyatkin és szerzőtársai (2017), (5) Kondor (2022a). A kérdőjelekkel jelölt eredmények ismereteink szerint nyitott problémák.

	Maximális súlyú m-dimenziós párosítás (MAX-mDM)		
	általános	pMAX-mDM	
	eset	$p = 1$	$p \geq 2$
$m = 2$	polinomiális (Edmonds, 1965b)		
$m \geq 3$	NP-nehéz (Feo és Khellaf, 1990)	?	NP-nehéz (Kondor, 2022a)

3.2. táblázat. Nehézségi eredmények a maximális súlyú m-dimenziós párosítás problémára és annak speciális eseteire. Az m a csoportok méretét, a p pedig az ℓ_p norma paraméterét jelöli. Forrás: Kondor (2022b).

		Minimális súlyú m-dimenziós párosítás (MIN-mDM)			
		általános	pMIN-mDM		
		eset	$p = 1$	$p = 2$	$p \geq 3$
$m = 2$		polinomiális (Edmonds, 1965b)			
$m = 3$		NP-nehéz (5)	NP-nehéz (5)	NP-nehéz (4)	NP-nehéz (5)
$3 < m < n/2$		NP-nehéz (5)			
$m = n/2$	$d = 1$	?	?	polinomiális (1)	?
	$d = 2$	?	?	polinomiális (2)	?
	d általános	NP-nehéz (5)	NP-nehéz (5)	NP-nehéz (3)	NP-nehéz (5)

3.3. táblázat. Nehézségi eredmények a minimális súlyú m-dimenziós párosítás problémára és annak speciális eseteire. Az m a csoportok méretét, a d az euklideszi tér dimenzióját, a p pedig az ℓ_p norma paraméterét jelöli. Forrás: Kondor (2022b).

4. fejezet

Egyoldali párosítási piacok

A közgazdászok által vizsgált egyik terület, hogy a közösségek hogyan allokálják az erőforrásokat. Némely allokációs problémára az árrendszer nyújt megoldást, ugyanakkor vannak olyan esetek, amelyekben ennek alkalmazása jogi vagy etikai akadályokba ütközne. Ilyen például az iskolai felvételi helyek gyerekekhez rendelése, vagy az emberi szervek transzplantációra várakozókhoz társítása. Továbbá van számos olyan piac is, amelyen ugyan működik árrendszer, viszont a tökéletes verseny hagyományos feltételei közel sem teljesülnek, mert például a javak oszthatatlanok és heterogének, így a piac az egyes árutípusok esetében nagyon szűk. Azt, hogy ezeken a piacokon milyen allokációk, vagy másképpen párosítások jönnek létre, az határozza meg, hogy milyen mechanizmusok szabályozzák azokat. (Nobel Prize, 2012b)

A párosítási piacok esetében, ha a piacon két különböző típusú szereplő található, akkor kétoldali párosítási piacról beszélünk. Erre ad példát az egyetemi felvételi eljárás, amelyben hallgatók és egyetemek között kell párosításokat kialakítani. Az egyoldali párosítási piacokon ugyanakkor csak egy típusú szereplő van, akiket párokba, vagy nagyobb csoportokba kell rendezni. A következőekben az utóbbihoz kapcsolódóan foglalkozunk a vesecsere-transzplantációk problémájával és hallgatók kollégiumi szobákhoz rendelésével.

A párosítási piacok vizsgálatának alapjait Gale és Shapley (1962) fektették le *stabilitásra* épülő megoldási koncepciójukkal. A 4.1. alfejezetben elsőként a stabil szobatárs problémát, és az ehhez kapcsolódó feladatok nehézségi eredményeit mutatjuk be. Ezt követően a 4.2. alfejezetben Segev és szerzőtársai (2005) tanulmánya nyomán egy alternatív megközelítést tárgyalunk a csoportok kialakítására. Ez egy m-dimenziós párosítási probléma, amely Pareto-hatékony megoldást ad eredményül. Ennek speciális eseteként

megadjuk az m-szobatárs probléma formális definícióját, amely a disszertáció későbbi fejezeteiben további vizsgálatok tárgyát képezi. Bemutatjuk az említett megközelítések nehézségi eredményeit, és végül a fejezetet a bemutatott főbb modellek előnyeinek és hátrányainak tárgyalásával zárjuk. A fejezet alapjául a Kondor (2022b) publikáció szolgál.

4.1. Stabil szobatársak probléma

Gale és Shapley (1962) felírja a *házasság problémát* (marriage problem), amely egy olyan párosítási feladat, ahol az egyének két csoportja van megadva, minden egyénnek szigorú preferenciarendezése van a másik csoport tagjain, és a feladat olyan párok létrehozása, amelynek tagjai különböző csoportokból származnak. Emellett a szerzők megadják a kapcsolódó *szobatárs problémát* (roommate problem), amelyben minden egyén egyazon csoportba tartozik, mindenki egyértelműen tud rangsorolni mindenki mást az alapján, hogy kivel mennyire szívesen kerülne egy párba és végül bárki bárkivel alkothat egy párt. (Biró és szerzőtársai, 2016)

A megfogalmazott problémákban a párosítások kialakítására Gale és Shapley (1962) a stabilitás fogalmát javasolja. Ennélfogva a problémákat gyakran stabil házasság, illetve stabil szobatárs problémának is nevezik (a problémákról magyarul lásd Biró (2006)). Azt mondjuk, hogy egy párosítás *stabil*, ha nincs két olyan, különböző párokban lévő résztvevő, akik jobban preferálják egymást, mint a saját párjukat. Utóbbi esetet nevezzük *blokkoló párnak* (blocking pair), így egy párosítás stabil, ha nincs blokkoló pár.[1] A stabilitás lényege tehát, hogy amennyiben a szereplők szabadon alakíthatnak ki új párokat, egy stabil megoldás esetében erre nem lesz motivációjuk, hiszen nem tudnak olyan új párt kialakítani, amelynek tagjai közül legalább egyvalaki jobban jár, és az új pár többi tagja közül senki sem jár rosszabbul. A következőekben a stabil szobatársak problémára vonatkozó eredményeket mutatjuk be mind párok, mind többfős csoportok esetében.

4.1.1. Stabil szobatársak párokra

Szigorú preferenciák esetében a szobatárs problémára - a házasság problémával ellentétben - stabil párosítás nem mindig létezik. Erre az egyik legegyszerűbb példa a következő. Legyen A, B, C és D négy hallgató, és preferencia-sorrendjeik legyenek a következőek:

[1] A stabil szobatársak problémára számos további megoldási koncepció is született az irodalomban, amelyek a jelen tárgyaláshoz kevésbé kapcsolódnak, lásd például Biró és szerzőtársai (2016).

A - 1: B, 2: C, 3: D

B - 1: C, 2: A, 3: D

C - 1: A, 2: B, 3: D

D - 1: A, 2: B, 3: C

Ekkor A, B és C mindegyike legjobban preferált a másik két hallgató közül valaki által. Így, ha tekintünk egy párosítást, akkor bárki is kerül D-vel egy szobába, a másik szobában lévő két hallgató közül az egyik biztosan jobban preferálja majd, mint a saját szobatársát, és mivel a D az A, a B és a C listáján is a legkevésbé preferált, ez a párosítás biztosan nem lesz stabil, sőt egyik sem.

Irving (1985) megadott egy olyan polinomiális, $\mathcal{O}(n^2)$ futásidejű algoritmust, amely a szobatárs probléma bármely esetére megmondja, hogy létezik-e stabil párosítás, és ha létezik, akkor meg is találja azt.

A megoldhatóság karakterizációját Tan (1991) adta meg, aki szükséges és elégséges feltételt mutatott egy stabil szobatárs létezésére. Tanulmányában a *stabil partícióval* általánosítja a stabil párosítás keretet, és megmutatja, hogy a stabil szobatárs probléma bármely esetére akkor és csak akkor létezik teljes stabil párosítás, hogyha nincs páratlan partíció.

Hogyha a preferenciarendezésekben megengedettek a döntetlenek, vagy másképpen indifferenciák is, akkor *gyenge preferenciákról* beszélünk. A döntetlenek bevezetése több stabilitási definíciót is maga után von. Azt mondjuk, hogy egy párosítás *gyengén stabil* (weakly stable), hogyha nincs olyan blokkoló pár, amelynek tagjai szigorúan jobban preferálják egymást saját partnerüknél. Továbbá egy párosítás *szuper-stabil* (super-stable), hogyha nincs olyan blokkoló pár, amelynek tagjai szigorúan jobban preferálnák a másikat saját párjuknál, vagy indifferensek közöttük.

A gyenge preferenciákra Ronn (1990) belátja, hogy a stabil szobatárs problémára NP-teljes feladat eldönteni, hogy egy adott esetre létezik-e gyengén stabil párosítás. Chung (2000) szintén a gyenge preferenciák esetét tekinti, és meghatározza a „nincsen páratlan hosszú kör” tulajdonságot, amely elégséges feltétele a gyengén stabil szobatárs párosítás létezésének.

A szuper-stabil párosításokra pozitív eredményként Irving és Manlove (2002) megad egy olyan lineáris futásidejű algoritmust, amely megtalál egy szuper-stabil párosítást, hogyha az létezik. Ezt az algoritmust továbbá kiterjesztik nem teljes és/vagy részbenren-

dezett preferencia-listák esetére is.

Emellett Irving és Manlove (2002) azt is megmutatja, hogy a stabil szobatárs probléma egy olyan esetére, amelyben döntetlenek és nem teljes preferencia-listák is megengedettek, több, különböző méretű gyengén stabil megoldás is előfordulhat. Belátják, hogy a legnagyobb méretű gyengén stabil párosítás megtalálása NP-nehéz feladat, ugyanakkor 2-es approxmációs rátával közelíthető.

Arkin és szerzőtársai (2009) a stabil szobatársak probléma egy olyan felírását tekintik, amelyben a hallgatókat pontok reprezentálják egy metrikus térben, és a preferencialistájukat a többi hallgatótól vett euklideszi távolságok sorrendje adja. Ezt *geometriai stabil szobatársak* (geometric stable roommates) problémának nevezik. Felhívjuk rá a figyelmet, hogy amíg az eredeti felírásban a preferencia-listák teljesen függetlenek egymástól, addig itt a konstrukcióból adódóan egymással összefüggenek. A szerzők megmutatják, hogy ennek a felírásnak a gyengén-stabil párosításra van polinomiális futásidejű megoldása még akkor is, hogyha döntetlenek is szerepelnek a preferencia-listában.

Emellett Arkin és szerzőtársai (2009) bevezetik az α-stabil párosítás fogalmat, amelynek értelmében a hallgatók csak akkor hajlandóak cserélni, hogyha legalább α-szoros javulást érnek el. Belátják, hogy egy α-stabil párosítás megtalálása legalább olyan nehéz, mint egy olyan párosítás megadása, amely a hagyományos értelemben stabil.

4.1.2. Stabil szobatársak magasabb dimenzióban

A gyakorlatban találhatóak olyan alkalmazások, amelyekben kétfősnél nagyobb csoportok kialakítása a feladat. Például a vesecserék esetében a transzplantációk az amerikai NEPKE programban három hosszú körökben is megengedettek (Biró, 2006). A kollégiumi hallgatókat pedig sok esetben 3 vagy akár 4-fős szobákba kell beosztani. A kétfősnél nagyobb párosítások kialakítására a stabilitás kontextusában adott megoldásokat a következőekben tárgyaljuk.

A stabil szobatárs probléma természetes módon adódó kiterjesztése a *3-dimenziós stabil szobatárs* (3-dimensional stable roommates, 3D-SR) probléma, amelyben háromfős szobák kialakítása a cél.[2] Ng és Hirschberg (1991) megmutatja, hogy ha az egyének a többi hallgatóból alkotott valamennyi párt rangsorolják, akkor a feladat NP-teljes. Meg-

[2]A 3-dimenziós stabil szobatárs problémát eredetileg Knuth (1977) vetette fel a *stabil házasságokról* (stable marriages) szóló könyvében az általa megfogalmazott 12 lehetséges további kutatási irány egyikeként.

oldásuk során a preferenciarendezések nem konzisztensek, ami azt jelenti, hogy egy e egyén preferencia-sorrendjében ($\succeq_e$) előfordulhat, hogy egy másik egyén rangsorolása attól függ, hogy kivel van párban. Például az x_1, x_2, y_1, y_2 egyénekre $x_1y_1 \succeq_e x_1y_2$ és $x_2y_2 \succeq_e x_2y_1$, ahol xy az x és y egyénekből alkotott párt jelöli.

Huang (2007) belátja, hogy a 3D-SR probléma előző, páros rangsorolásos megfogalmazása akkor is NP-teljes, hogyha a preferenciarendezések konzisztensek, azaz $xy \succeq_e xz$ vagy minden x egyénre teljesül, vagy egyikre sem. Konzisztens preferenciarendezésre példaként egy olyan felírást említenek, amelyben az egyének értékeléseket adnak a többiekre, és a párok értéke a benne szereplő egyének értékeinek összege. A kombinációk sorrendjét az értékek összege határozza meg, és amennyiben két különböző párra az értékek összege egyenlő, úgy az egyén indifferens közöttük.

Iwama és szerzőtársai (2007) bizonyítják, hogy a 3D-SR probléma NP-teljes még akkor is, hogyha a hallgatók nem párokat rangsorolnak, hanem minden más jelentkezőt különkülön.

Deineko és Woeginger (2013) definiálja a 3D-SR probléma egy metrikus térbeli megközelítését (Metric-3D-SR), amelyben az egyének pontoknak felelnek meg egy metrikus térben, akik között egy sztenderd távolságfüggvényt értelmezünk. Egy e egyén preferenciarendezése a párok fölött van megadva, a rangsort pedig az e egyéntől vett távolságok összege határozza meg. Vegyük észre, hogy ez a Huang (2007) által megadott példa egy speciális esete. A szerzők belátják, hogy a probléma ezen felírása NP-teljes.

Chen és Roy (2021) a Metric-3D-SR probléma speciális eseteként megmutatja, hogy az NP-teljesség akkor is fennáll, hogyha az egyéneket euklideszi térbeli pontokként reprezentáljuk (így definiálva az Euclid-3D-SR problémát), és a távolságfüggvényt a pontok közötti távolság adja.

Arkin és szerzőtársai (2009) a stabil szobatárs probléma 3-dimenziós változatát is tekintik, amelyre geometriai 3D-SR problémaként hivatkoznak. A szerzők bizonyítják, hogy a 3-dimenziós geometriai stabil szobatársak problémára egy 2-stabil párosítás polinomiális futásidőben megadható.

A 3D-SR probléma további variánsaiként Biró és McDermid (2010) belátja a 3-dimenziós probléma NP-teljességét ciklikus preferenciákra nem teljes listák esetén, míg Lam és Plaxton (2019) bizonyította az NP-teljességet teljes ciklikus listákra, amely eredményt kiterjesztették 3-nál magasabb dimezióra is.

4.2. Az m-szobatárs probléma, mint egyenletes klaszterezési megközelítés

Morrill (2010) a szobatárs probléma megoldása során abból indul ki, hogy a valóságban, ha már kialakult egy beosztás, akkor hiába van a párosításban blokkoló pár, annak tagjai nem kényszeríthetik saját szobatársaikat, hogy kiköltözzenek, valamint ha nincsen üres szoba sem, az új pár nem tud egyoldalúan létrejönni. Ez csupán akkor valósulhat meg, hogyha van olyan koalíció, amelynek tagjai úgy tudnak új párokat kialakítani, hogy egyik szereplő sem jár rosszabbul. Így a kapott koalíciós játékban az általa javasolt megoldási koncepció a Pareto-hatékonyság.

A következőekben egy olyan megközelítést adunk meg, amely az előzőekkel összhangban Pareto-hatékony megoldást határoz meg. Ehhez Segev és szerzőtársai (2005) tanulmányát vesszük alapul, akik a vesecserék esetében a párok kialakítására az "első találatot elfogadó" eljárással szemben egy optimalizált párosítási módszert javasolnak. Ez az általuk elvégzett szimulációs kísérlet alapján országos szinten több transzplantációt, jobb HLA egyezőséget (a transzplantációk során különösen fontos tényező), az átültetett szervek jobb 5 éves várható túlélését, és az utazásra kényszerülő párok számának csökkenését eredményezné. A javasolt eljárást alkalmazták Ohio államban, és matematikailag egy súlyozott párosítás feladatra vezethető vissza (Biró, 2006).

A súlyozott párosítási megközelítést magasabb dimenzióban az m-dimenziós párosítás problémával, pontosabban egy maximalizálási célra is értelmezett egyenletes klaszterezési feladatként általánosítjuk. Vegyük észre, hogy a vesecsere-transzplantációk esetében a kialakított csoportokon belül nem az egyének közötti minden élt kell figyelembe venni, hanem gyakorlatilag körök kialakítása a feladat, így erre az alkalmazásra csak legfeljebb $m = 3$-ra tekinthetjük ezt a felírást. Emellett a sajátosságaiból eredően a vesecsere problémára célszerűbb lenne olyan modellt választani, amely figyelembe tudja venni, hogy egy hármas csoporton belül egy beteg-donor pár hasznossága a két másik szereplő együttes függvénye, valamint megengedi egyszerre a két- és háromfős csoportokat is. Ezen megfontolások miatt az általánosítást a szobabeosztások kontextusában tárgyaljuk.

Az m-dimenziós párosítás probléma (minimalizálási esetben MIN-mDM, valamint maximalizálási esetben MAX-mDM) gráfparticionálási problémaként tekinti a csoportosítási feladatot. Az egyszerűbb tárgyalás érdekében a következőekben mi egy maximalizálási

célra is értelmezett egyenletes klaszterezési feladatként kezeljük azt. A minimalizálási verziót a 2MIN-mDM, míg a maximalizálási verziót a 2MAX-mDM optimalizálási problémával (lásd a 3.3.29. Optimalizálási problémát) írjuk le. A szobatárs kontextusban megfogalmazott feladatokra egységesen *m-szobatárs problémákként* hivatkozunk.

4.2.1. Az m-szobatárs probléma formális definíciója

Az előzőeknek megfelelően az m-szobatárs probléma formális definíciója a következő. Legyen n a felvételt nyert hallgatók száma, és legyen m egységesen a szobánkénti férőhelyek száma. Az egyszerűség kedvéért legyen n az m egész többszöröse, a szobák száma pedig ennél fogva legyen $k = n/m$. A szobákat jelölje $C_1, \ldots, C_k$. Tegyük fel továbbá, hogy azt, hogy két ember mennyire lenne megfelelő szobatárs, bizonyos tulajdonságok alapján döntjük el a következő módon. Legyen d a *tulajdonságok* száma, és tegyük fel, hogy a tulajdonságok tetszőleges valós értékek lehetnek.[3] Ennek következtében egy hallgató megfeleltethető egy $\mathbb{R}^d$-beli koordinátának, és az összes hallgató elhelyezhető egy d-dimenziós euklideszi térben. Jelölje $x_1, \ldots, x_n \in \mathbb{R}^d$ a hallgatókat és legyen $D \in \mathbb{R}^{d \times d}$ a hallgatók kölcsönös távolságainak mátrixa. Jelölje továbbá $x_i \in C_s$, ha az i-edik hallgatót az s-edik szobába osztottuk be.

Feltesszük, hogy azt, hogy az x_i és x_j hallgatók mennyire lennének jó szobatársak egymás számára a közöttük lévő euklideszi távolság, vagyis a $d_{ij} = \|x_i - x_j\|$ érték reprezentálja. A *jó* szobabeosztásra két különböző megközelítést tekintünk.

Az egyik során a hasonló tulajdonságokat, vagyis a pontok közötti kisebb távolságokat tekintjük jónak, és a lehető leghomogénebb szobabeosztásokat keressük. Emögött az az elképzelés áll, hogy a hasonló tulajdonságokkal rendelkező hallgatók könnyebben alakíthatnak ki barátságokat, hosszú távon megfelelőbb szobatársak lehetnek egymásnak. Ekkor a tulajdonságok reprezentálhatnak akár szakokat, vagy érdeklődési területeket, például hogy ki mennyire szeret egy-egy zenei műfajt vagy hobbit. Így tehát egy minimalizálási problémát adunk meg:

$$\min \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} d_{ij}^2, \qquad \text{(2MIN-}m\text{DM)}$$

$$\text{s.t. } |C_s| = m \ \forall s.$$

[3] Az elemzés során az egyszerűség kedvéért azt tesszük fel, hogy a tulajdonságoknak megfelelő értékek a $[0, 10]$ intervallumból vehetnek fel egész értékeket.

A másik megközelítés a napjainkban egyre nagyobb figyelmet kapó diverzitással járó előnyöket hivatott megragadni. E során a hallgatók közötti nagyobb távolságokat tekintjük jónak, és a lehető legváltozatosabb, vagy másképpen legheterogénebb csoportokat szeretnénk kialakítani. A megjelenített tulajdonságok pedig lehetnek szakok, szakirányok, életkor, korábbi tanulmányok, stb. Az adódó maximalizálási problémát a

$$\max \sum_{s=1}^{k} \sum_{x_i, x_j \in C_s} d_{ij}^2, \qquad \text{(2MAX-}m\text{DM)}$$
$$\text{s.t. } |C_s| = m \ \forall s$$

alakban írjuk fel.

A megegyező célfüggvények garantálják, hogy egy egységes modell keretein belül vizsgálhassuk az m-szobatárs probléma minimalizálási és maximalizálási verzióit. A célfüggvény a minimalizálási problémában a már megszokott, MSSC feladatban is szereplő. Ugyanakkor az irodalomban a maximalizálási problémák során hagyományosan - ismereteink szerint - a négyzetes távolságösszegek helyett egyszerű távolságösszegeket tekintenek, így ez szokatlan választásnak hathat. Vegyük észre, hogy ezzel a felírással a csoporton belüli kívülálló egyének szerepét erősítjük, cserében pedig a tulajdonságokat szélesebb skálán reprezentáljuk a csoportokon belül.

4.2.2. A modell előnyei és hátrányai

Elméleti szempontból az egyik legfontosabb különbség a stabil szobatárs és az m-szobatárs problémák között, hogy amíg előbbi stabil megoldást ad eredményül, addig utóbbi egy Pareto-hatékony megoldási keretet határoz meg. Az utóbbi által adott megoldás tehát nem feltétlenül stabil, viszont bizonyos esetekben jobb választás lehet a gyakorlati probléma leírására (lásd Morrill (2010)).

A stabilitási koncepció esetében megmutattuk, hogy a kétfős csoportokra előfordulhat, hogy a megoldások halmaza az üres halmaz, továbbá gyenge preferenciák esetében annak eldöntése, hogy létezik-e gyengén stabil párosítás, NP-teljes feladat. A megközelítés magasabb dimenziós eredményeinél pedig láthattuk, hogy egy kivétellel az összes felírás csupán a háromfős csoportokra vonatkozik, és szintén egy kivétellel az összes megfogalmazás NP-teljes feladatra vezet.

Ezzel szemben az m-szobatárs problémában, optimalizálási problémáról lévén szó, min-

dig van megoldás. Emellett párok, vagyis $m = 2$ esetén Edmonds (1965b) algoritmusa révén mindig meg tudjuk adni a megoldást polinomiális futásidőben. Legalább háromfős csoportokra ugyanakkor már ennél a megközelítésnél is nehézségekkel találjuk szembe magunkat. A 3.3.30. és 3.3.33. Tételek alapján tudjuk, hogy a 2MIN-mDM, valamint a 2MAX-mDM problémák NP-nehezek. Ennek értelmében pedig, habár tudjuk, hogy bármely klaszterméretre létezik optimum, azt a nagyobb méretű problémák, vagyis a hallgatók nagyobb száma mellett képtelenek vagyunk meghatározni (Kondor, 2022b).

Gyakorlati szempontból az m-szobatárs szerinti felírásnak - egyúttal az egyéb metrikus térbeli megközelítéseknek is - az az előnye a stabil szobatársak problémával szemben, hogy nincs szükség arra, hogy a hallgatók egyéni preferencia-sorrendeket adjanak meg. Ez olyan esetekben jelent reálisabb megközelítést a szobabeosztásokra, amikor vannak olyan hallgatók, akiknek nagy valószínűséggel nincs semmilyen preferencia-sorrendje a többiekre nézve. Ez a helyzet fordul elő a kollégiumi felvételek során, amikor sokaknak jellemzően nincs információja a többi hallgatóról, és így rangsort sem tudnak felállítani közöttük.

A modell további pozitívuma a preferencia-listák elhagyásában rejlik. Ugyanis, a preferencia-sorrendek esetén nincs mértékbeli különbség a rangsor elemei között. Vagyis ha egy x hallgató rangsorában y és z egyénekre vonatkozóan $y \succ_x z$ szerepel (vagyis x szempontjából y jobb, mint z), akkor nem tudjuk, hogy y mennyivel jobban preferált z-vel szemben. Ezzel ellentétben a távolságok többletinformációt adnak, így a gyakorlatban hozzájárulhatnak egy jobb szobabeosztás kialakításához.

Az m-szobatárs modell egyik negatívuma a stabil szobatársak problémával szemben, hogy míg utóbbiban az y és z hallgatóknak lehetett eltérő preferenciája a másikról, addig előbbiben a távolság révén a szerepük szimmetrikus.

A felírás másik hátránya, hogy hiányzik az egyéni preferenciák megadásának lehetősége. Elképzelhető ugyanis, hogy két vagy több hallgató, akik már ismerik egymást, mindenképpen ugyanabba a szobába szeretnének kerülni. Ahhoz, hogy ezt figyelembe tudjuk venni, egy általánosabb modell, például az m-dimenziós párosítás alkalmazása lenne a megoldás, amelyben a hallgatók között tetszőleges élsúlyokkal egyéni távolságokat is megadhatunk. Ebben az esetben a minimalizálási problémára akár olyan élsúlyokat is definiálhatunk, amely révén két hallgató biztosan különböző szobákba kerül, amelyre egyik előzőleg említett modell sem alkalmas. Az általános modell további tárgyalása ugyanakkor nem célja jelen dolgozatnak, de érdekes további kutatási irány lehet.

5. fejezet

Garanciák a szuboptimalitásra

Az m-szobatárs probléma legalább 3-fős szobák esetén NP-nehéz, így - jelen ismereteink szerint - nem létezik olyan algoritmus, amely polinomiális futásidőben megadná az optimumot. Emiatt általánosan nincs esélyünk arra, hogy hatékonyan megtaláljuk az optimális megoldást.

A fejezet során olyan algoritmusokat tekintünk, amelyek egy meghatározott egyenletes klaszterezési feladatra megadnak egy megengedett megoldást, és valamilyen garanciát is nyújtanak annak (szub)optimalitására. Az approximációs eljárások olyan problémákra vonatkoznak, amelyekben a célfüggvényben egyszerű távolságösszegek szerepelnek. Ezek teljesítményi aránya a megengedett megoldás és az optimum hányadosára ad meg egy korlátot.

A kúp optimalizálási megközelítés pedig az elemszámkorlátokkal ellátott MSSC problémára vonatkozik, amelyben négyzetes távolságösszegek jelennek meg. E során a relaxált probléma megoldása, és az annak helyreállításával kapott megengedett megoldás alsó és felső határokat nyújtanak az optimum értékére.

5.1. Approximációk

A fejezet során approximációs eljárásokat tekintünk, amelyek futásideje polinomiális, és egy meghatározott r teljesítményi aránnyal, vagy másképpen approximációs rátával rendelkeznek. Ezek garantálják, hogy az általuk talált megoldás és az optimum között legfeljebb r-szeres az eltérés, ennél rosszabb megoldást nem kaphatunk. Ugyanakkor sok esetben az approximációs ráta éles, vagyis található olyan példa, ahol az approximációs

algoritmus által megadott megoldás célfüggvényértéke az algoritmus teljesítményi rátájának megfelelő, vagy ahhoz nagyon közeli. Ez a gyakorlatban egy 2-höz közeli teljesítményi arány esetén még viszonylag laza megszorítást jelent.

Az r-approximációs algoritmusok 3.1.27. Definíciójától eltérően a fejezet során megkülönböztetjük a minimalizálási és maximalizálási problémákat. Egy minimalizálási probléma során az $r \geq 1$ teljesítményi arány esetében r-approxmációs eljárásról, míg maximalizálási probléma esetén $1/r$-approximációs algoritmusról beszélünk.

A fejezet során egységesen a következő jelöléseket használjuk. Legyen $G = (V, E)$ gráf esetén $w : E \to \mathbb{R}$ az éleken értelmezett függvény, amely egy $e \in E$ élre megadja az él $w(e)$ súlyát vagy élhosszát. Legyen az élek egy $E' \subset E$ részhalmazára $w(E') = \sum_{e \in E'} w(e)$. Jelölje a csúcsok egy $V' \subset V$ részhalmazára $E(V') = \{e | e = (u, v) \in E, u, v \in V, u \neq v\}$ a V' által indukált részgráfot, és legyen $w(V') = w(E(V')) = \sum_{e \in E(V')} w(e)$ a V' által indukált részgráf éleinek összköltsége.

Jelölje továbbá *Apx* minden esetben a szóban forgó optimalizálási problémára a megadott approxmációs eljárás által talált megengedett megoldást, míg *Opt* jelölje a feladat optimális megoldását. Végül jelölje $w(Apx)$ és $w(Opt)$ rendre ezen megoldások célfüggvényértékét a megfogalmazott optimalizálási probléma szerint.

5.1.1. k-particionálás probléma

Feo és Khellaf (1990) az NP-teljességi bizonyításon felül approximációs eljárásokat is megad a maximális súlyú m-dimenziós párosítás, vagy ekvivalensen a k-particionálás problémára, pontosabban annak maximalizálási verziójára. Algoritmusaik a maximális súlyú teljes párosításon alapulnak és 1/3-approximációkat adnak meg $k = |V|/3$, valamint $k = |V|/4$ esetére.

Tekintsük a k-particionálás problémát, ahol a kívánt klaszterenkénti csúcsok száma m. Legyen adott egy $G = (V, E)$ irányítatlan, teljes gráf, amelynek csúcshalmazára $|V| = km$ teljesül, ahol k pozitív egész. Feo és Khellaf (1990) két külön esetet kezel aszerint, hogy m páros, vagy páratlan. Ennek megfelelően a P1 (ha m páros), illetve P2 (ha m páratlan) eljárásuk a csúcsokat k darab, egyenként m csúcsot tartalmazó halmazra particionálja. Az eljárások pszeudokódjait az 5.1. és 5.2. Algoritmusok írják le.

P1 eljárás (Feo és Khellaf, 1990)

Input Egy $G = (V, E)$ teljes gráf, $|V| = km, m$ páros, $w(e) \geq 0\ \forall e \in E$ élre

(1) $M \leftarrow$ maximális súlyú teljes párosítás G-ben

(2) **for** $i = 1, \ldots, k$

Válasszuk ki az M nem megjelölt éleinek egy $m/2$ elemű, π halmazát

Jelöljük meg π éleit

Helyezzük az i-edik klaszterbe a π éleinek végpontjait

end

5.1. Algoritmus. P1 eljárás (Feo és Khellaf, 1990)

P2 eljárás (Feo és Khellaf, 1990)

Input Egy $G = (V, E)$ teljes gráf, $|V| = km, m$ páratlan, $w(e) \geq 0\ \forall e \in E$ élre

(1) $M \leftarrow (m-1)|V|/(2m)$ élet tartalmazó, maximális súlyú teljes párosítás G-ben

(2) **for** $i = 1, \ldots, k$

Válasszuk ki az M nem megjelölt éleinek egy $(m-1)/2$ elemű, π halmazát

Jelöljük meg π éleit

Helyezzük az i-edik klaszterbe a π éleinek végpontjait, és bármely olyan nem M-beli csúcsot, amely még nem eleme valamely klaszternek

end

5.2. Algoritmus. P2 eljárás (Feo és Khellaf, 1990)

Az algoritmusok approximációs eredményeit az alábbi tételek mondják ki.

5.1.1. Tétel. *Ha az élsúlyokra teljesül a háromszög-egyenlőtlenség, akkor az m-dimenziós párosítás probléma optimális megoldásának értéke legfeljebb $2(m-1)/m$-szerese a P1 eljárás által megadott megoldás értékének. A korlát az általános esetben $(m-1)$.*

Amennyiben teljesül a gráf élsúlyaira a háromszög-egyenlőtlenség, a fenti tételben szereplő hányados értéke $m = 4$ csúcsból álló klaszterek esetén $3/2$, tehát egy $2/3$-

approximációt ad meg. Ugyanez abban az esetben, amikor a háromszög-egyenlőtlenség nem áll fenn, egy 1/3-approximációt eredményez.

5.1.2. Tétel. *Ha az élsúlyokra teljesül a háromszög-egyenlőtlenség, akkor az m-dimenziós párosítás probléma optimális megoldásának értéke legfeljebb $2m/(m+1)$-szerese a P2 eljárás által megadott megoldás értékének. A korlát az általános esetben m.*

Három elemű partíciók ($m = 3$) és háromszög-egyenlőtlenség esetén az előző tétel egy 2/3-approximációt jelent. Ha a háromszög-egyenlőtlenség nem áll fenn, akkor 1/3-aproximációról beszélhetünk.

Látható az is, hogy az approximációs ráták annál szigorúbb korlátot adnak meg, minél kisebb az m értéke, vagyis minél kisebb méretű csoportokat kívánunk létrehozni. Feo és Khellaf (1990) egy további heurisztikát is megad a $|V|/4$-particionálás problémára, amely első lépésben egy maximális súlyú teljes párosítást keres, majd elvégez egy összevonó lépést, majd ezután egy újabb maximális súlyú teljes párosítás következik. Ennek pszeudokódját nem közöljük, egy általánosabb verzióját Feo és szerzőtársai (1992) Párosítás-és-összevonás (lásd 5.3. Algoritmus) heurisztikája írja le.

Feo és szerzőtársai (1992) szintén a k-particionálás problémát tekinti, ahol az $m = 3$ és $m = 2^r$ esetek kapnak kiemelt figyelmet. Megkonstruálnak egy $|V|/3$-particionáló algoritmust, amely azon az ötleten alapszik, hogy egy $|V|/3$-partíció egy olyan teljes 2-párosítás, amely 3-hosszú körökből áll.

5.1.3. Definíció. Reguláris gráf, teljes b-párosítás
Azt mondjuk, hogy egy $G = (V, E)$ gráf reguláris*, hogyha minden csúcsának ugyanannyi szomszédja van, vagy másképpen, minden csúcs fokszáma azonos. A G gráf egy* teljes b-párosítása *az E éleinek egy olyan E' részhalmaza, amelyre a $G = (V, E')$ gráf reguláris b fokszámmal.*

Ennek következtében, egy maximális súlyú teljes 2-párosítás súlya nagyobb vagy egyenlő, mint egy optimális $|V|/3$-partíció súlya. A konstrukció lényege, hogy első lépésként egy maximális súlyú 2-párosítást keresnek, majd az ebből adódó megoldásból alkotnak 3-élhosszúságú köröket a 3-nál több élt tartalmazó körök felbontásával és átrendezésével. Az algoritmus egy 1/2-approximáció:

5.1.4. Tétel. *A heurisztikus eljárás által kapott $|V|/3$-partícióban a csoporton belüli élek teljes súlya legalább feleakkora, mint az optimális megoldás csoporton belüli élsúlyainak összege.*

Feo és szerzőtársai (1992) arra a k-particionálás problémára, amikor m értéke a 2 valamely hatványa, Feo és Khellaf (1990) *párosítás-és-összevonás* (match-and-contract) heurisztikáját alkalmazzák. Ennek pszeudokódját az 5.3. Algoritmus írja le.

Párosítás-és-összevonás (match-and-contract) heurisztika (Feo és szerzőtársai, 1992)	
Input	Egy $G = (V, E)$ teljes gráf, $\|V\|/k = m, m = 2^r$, nemnegatív élsúlyokkal
(1)	$G' \leftarrow G$
(2)	**do** (r-szer) Találjunk egy maximális súlyú teljes párosítást G'-ben Legyen G' az a gráf, amelyet úgy kapunk, hogy a) összevonjuk a párosításban lévő éleket, és b) összevonjuk az így kapott párhuzamos éleket az élsúlyaik összeadásával **end**
(3)	Képezzük a k darab, egyenként $m = 2^r$ csúcsból álló klaszter mindegyikét a G'-ben lévő k darab összevont csúcs egyenkénti felbontásával.

5.3. Algoritmus. Párosítás-és-összevonás (match-and-contract) heurisztika (Feo és szerzőtársai, 1992)

5.1.5. Tétel. *A párosítás-és-összevonás heurisztika alkalmazásával kapott, valamint az optimális megoldásra $k = |V|/4$ esetén $w(Apx)/w(Opt) \geq 1/2$.*

A $|V|/3$-particionáló heurisztikában alkalmazott 2-párosítás és a párosítás-és-összevonás heurisztikában a kezdeti párosítás megtalálásának futásideje egyaránt $\mathcal{O}(|V|^3)$ (Papadimitriou és Steiglitz, 1982). Így a Feo és szerzőtársai (1992) tanulmányában tárgyalt mindkét módszer futásideje $\mathcal{O}(|V|^3)$.

5.1.2. Speciális elemszám megkötések

Hassin és szerzőtársai (1997) a *maximális szóródás* (maximum dispersion) problémára ad meg approximációs algoritmusokat. A problémát a következőképpen írják fel. Legyen adott egy $G = (V, E)$ irányítatlan teljes gráf, $V = \{v_1, \ldots, v_n\}$ csúcshalmazzal. A

$w(e), e \in E$ élsúlyokról feltesszük, hogy nemnegatívak, és hogy fennáll rájuk a háromszög-egyenlőtlenség. Legyen adott $p \in \{2, \ldots, n\}$ és $k \in \{1, \ldots, \lfloor n/p \rfloor\}$. A feladat, hogy találjunk V-ben k darab, $P_1, \ldots, P_k$ diszjunkt részhalmazt, amelyekre $|P_i| = p, i = 1, \ldots, k$ és $\sum_{i=1}^{k} w(P_i)$ maximális. Vegyük észre, hogy itt a csoportok k száma akár 1 is lehet.

Hassin és szerzőtársai (1997) algoritmusa	
(1)	Találjunk egy M^* maximális q-párosítást G-ben, ahol $q = k\lfloor p/2 \rfloor$ $V^* \leftarrow \{M^*$ éleinek végpontjai $\}$
(2)	Particionáljuk M^*-ot tetszőlegesen k darab, $M_1, \ldots, M_k$ részhalmazra, melyek mindegyike $\lfloor p/2 \rfloor$ élet tartalmaz $P_1, \ldots, P_k \leftarrow$ rendre az $M_1, \ldots, M_k$-beli élek végpontjainak halmazai
(3)	Ha p páratlan, egészítsük ki a megoldást úgy, hogy minden részhalmazhoz hozzáadjuk $V \backslash V^*$ egy tetszőleges diszjunkt csúcsát

5.4. Algoritmus. Hassin és szerzőtársai (1997) algoritmusa

Legyen egy *q-párosítás* egy q darab csúcs-diszjunkt élből álló halmaz. Legyen továbbá egy *maximális q-párosítás* egy maximális összsúlyú q-párosítás. Hassin és szerzőtársai (1997) metódusát, amely a csoportok bármely k száma esetén legalább 1/2-es garanciát ad, az 5.4. Algoritmus írja le.[1] Az algoritmusra vonatkozó approximációs eredményt a következő tétel fogalmazza meg.

5.1.6. Tétel. $(2 - 1/\lceil p/2 \rceil)\, w(Apx) \geq W(Opt)$.

A tételben szereplő zárójeles kifejezés értéke $p = 3$ esetén 3/2. Ez azt jelenti, hogy az algoritmus (a háromszög-egyenlőtlenség teljesülése mellett) három elemből álló klaszterekre egy 2/3-approximációt ad meg. Emellett nagyobb p értékek, vagyis a partíciók nagyobb elemszáma esetén az approximációs ráta kevésbé kedvező.

Az algoritmus futásidejét az első lépésben található maximális q-párosítás megtalálása dominálja. Jelölje $n = |V|$ a csúcsok, míg $m = |E|$ az élek számát. Arra alapozva, hogy egy maximális súlyú teljes párosítás $\mathcal{O}(n^3)$ időben megkonstruálható (lásd Gabow (1976)), erre Hassin és szerzőtársai (1997) megad egy $\mathcal{O}(n^3)$ futásidejű algoritmust. Ugyanakkor

[1] Hassin és szerzőtársai (1997) tanulmányukban megadnak egy második algoritmust is, amely $k = 1$ esetén ugyanazt a korlátot adja alacsonyabb futásidő mellett.

megjegyzik, hogy a konstrukció komplexitását lehet csökkenteni, ha kezdeti lépésként a $2q-1$ legnagyobb súlyú élen kívüli éleket töröljük, amely $\mathcal{O}(n^2)$ futásidőn belül elvégezhető (lásd Blum és szerzőtársai (1973)). Ekkor, felhasználva, hogy egy ritka gráfon egy maimális súlyú teljes párosítás $\mathcal{O}(nm + n^2 \log n)$ időben megtalálható (lásd Gabow (1990)), a konstrukció futásideje $\mathcal{O}(n^2(kp + \log n))$.

Hassin és Rubinstein (2006c) a *metrikus maximális súlyú klaszterezési problémát* tárgyalja *megadott klaszterméretekkel* (metric maximum clustering problem with given cluster sizes). A $G = (V, E)$ gráf élsúlyai nemnegatívak, és teljesül rájuk a háromszög-egyenlőtlenség. Legyenek előre megadva a $c_1, \ldots, c_p$ klaszterméretek, amelyekre álljon fenn a $\sum_{i=1}^{p} c_i \leq n$ egyenlőtlenség. Az általánosság megszorítása nélkül tegyük fel, hogy $c_1 \geq \cdots \geq c_p$.

Hassin és Rubinstein (2006c) konstrukciójának egyik alapja, hogy meghatározható maximális q-párosítások olyan növekvő számú csúcsból álló $M_j, j = 1, \ldots, \lfloor n/2 \rfloor$ sorozata, amelyre $V(M_j) \subset V(M_{j+1})$, ahol $V(M)$ jelöli az M párosításban lévő csúcsok halmazát. Másik alapja pedig, hogy *rétegeket* (layer) határoznak meg, amelyekhez a csoportokat hozzárendelik. Egy réteghez több csoport is tartozhat, valamint egy csoport több réteghez is tartozhat.

Az algoritmus pszeudokódját az 5.5. Algoritmus írja le. Jelölje $q_i = \lfloor c_i/2 \rfloor$ azt, hogy az i-edik csoport mennyi (nem egy csúcsot tartalmazó) réteghez tartozik. Ekkor a rétegek száma q_1, ezeket jelölje $L_1, \ldots, L_{q_1}$. Jelölje $I_j = \{i |\ q_i \geq q_1 - j + 1\}$ azon csoportok halmazát, amelyek a j-edik réteghez tartoznak. Az L_1 réteghez pontosan azok az i csoportok tartoznak, amelyekre q_i megegyezik q_1-gyel. Az algoritmus az (1)-es lépés egy iterációjában egy adott L_j réteghez tartozó valamennyi csoporthoz hozzárendel két elemet. A (2)-es lépésben az eljárás a páratlan elemszámú csoportokhoz rendel véletlenszerűen 1-1 csúcsot.

Hassin és Rubinstein (2006c) algoritmusa

Input Irányítatlan, teljes $G = (V, E)$ gráf, w metrika az E éleken
$c_1 \geq \cdots \geq c_p$ egészek, amelyekre $\sum_i c_i \leq |V|$

(0) $q_i \leftarrow \lfloor c_i/2 \rfloor, i = 1, \ldots, p$
$m_0 \leftarrow 0, M_{m_0} \leftarrow \emptyset$

(1) **for** $j = 1, \ldots, q_1$
$I_j \leftarrow \{i |\ q_i \geq q_1 - j + 1\}$ [a j-edik réteghez tartozó csoportok indexei]

$m_j \leftarrow m_{j-1} + |I_j|$

$M_{m_j} \leftarrow$ maximális m_j-párosítás, amelyre $V(M_{m_{j-1}}) \subset V(M_{m_j})$

$L_j \leftarrow V(M_{m_j}) \backslash V(M_{m_{j-1}})$

Az L_j-beli csúcsokat véletlenszerűen párokba rendezzük, és azokat véletlenszerűen szétosztjuk azon C_i csoportok között, amelyekre $i \in I_j$ úgy, hogy minden csoportba pontosan egy pár kerüljön

end

(2) A páratlan elemszámú csoportokhoz véletlenszerűen hozzárendelünk egy-egy különböző elemet a $V \backslash V(M_{m_{q_1}})$ halmazból

5.5. Algoritmus. Hassin és Rubinstein (2006c) algoritmusa

Hassin és Rubinstein (2006c) az algoritmusra az alábbi eredményt bizonyítják, amely aszimptotikusan 2-es approximációs rátát igazol.

5.1.7. Tétel. *Legyen $k = c_p > 6$. Az 5.5. Algoritmus egy $\left(\frac{1}{2} - \frac{3}{k}\right)$-approximációt ad meg.*

5.1.3. Maximális súlyú háromszög pakolás

A *maximális súlyú háromszög pakolás* (maximum-weight triangle packing, MWTP) a k-particionálás probléma speciális esete háromelemű partíciókkal. Hassin és Rubinstein (2006a) felírásában legyen $G = (V, E)$ teljes, irányítatlan gráf V csúcshalmazzal, amelyre $|V| = 3n$, valamint jelölje E az élek halmazát. Nevezzünk egy három élből álló kört háromszögnek. A feladat n darab csúcs-diszjunkt háromszög megtalálása, amelyek éleinek összsúlya maximális.

Hassin és Rubinstein (2006a,b) dolgozta ki az első olyan algoritmust, amelynek aprroximációs rátája szigorúan nagyobb, mint 1/2. Ennek teljesítménye $\frac{43}{83}(1-\varepsilon) \approx 0{,}518(1-\varepsilon)$ minden $\varepsilon > 0$ konstansra.[2]

Az eljárás első lépésként megkonstruál egy maximális súlyú 2-párosítást (a 2 a fokszámra utal), amely egyszerűbben fogalmazva, legalább három éllel rendelkező körökből álló fedés. Azon C köröket, melyekben az élek $|C|$ számára $|C| > \varepsilon^{-1}$ teljesül, olyan utakra bontja, amelyek legfeljebb ε^{-1} élhosszúságúak, majd ezeket körökre egészti ki, amely

[2] Az eredeti cikkben $(\frac{89}{169} - \varepsilon)$ szerepelt, de vétettek egy hibát az analízisükben, így módosították az eredményt.

végül megadja a $\mathscr{C}$, szintén körökből álló fedést (cycle cover). Ezután az algoritmus három különböző eljárással három megoldást konstruál meg, amelyek közül végül a legjobbat választja ki. Ezek eléggé technikaiak, így az algoritmusok pszeudokódját itt nem közöljük. Az első algoritmus determinisztikus, és akkor teljesít jól, amikor az *Opt* nagy része $\mathscr{C}$-beli élekből származik. A második szintén determinisztikus, és akkor eredményes, amikor az *Opt* nagy része olyan élekből ered, amelyek mindkét csúcsa ugyanazon a $\mathscr{C}$-beli körön található. A harmadik egy komplex véletlen algoritmus. Ez fedi le a fennmaradó eseteket, amikor a megoldás jelentős részét olyan élek adják, amelyek $\mathscr{C}$ diszjunkt köreit kapcsolják össze. Az algoritmus futásideje $\mathcal{O}(n^3)$.

Ezt vette alapul Chen és szerzőtársai (2009, 2010), és a harmadik, véletlen eljárás módosításával egy olyan polinomiális futásidejű, véletlen algoritmust konstruáltak, amellyel kicsivel jobb, $\approx 0{,}523(1-\varepsilon)$-approximációt kaptak eredményül minden $\varepsilon > 0$-ra.[3]

Van Zuylen (2013) *pesszimista becslők* (pessimistic estimator) felhasználásával belátja, hogy mindkét előző algoritmus derandomizálható, és így léteznek olyan determinisztikus algoritmusok, melyek a kapott approximációs rátával rendelkeznek. Ekképpen adódik a következő eredmény.

5.1.8. Tétel. *Létezik determinisztikus* $0{,}523(1-\varepsilon)$*-approximációs algoritmus a maximális súlyú háromszög pakolás problémára.*

Chen és szerzőtársai (2021) a *metrikus maximális súlyú háromszög pakolás* (metric maximum-weight triangle packing, MMWTP) problémát tekinti. Ebben a különbség az egyszerű MWTP problémához képest, hogy az élsúlyokra teljesül a háromszög-egyenlőtlenség. A szerzők egy nemtriviális, véletlen algoritmust adnak meg, amelynek felépítése az előzőekhez hasonló abban a tekintetben, hogy itt is 3 különböző megoldást konstruálnak, és ezek közül választják a legjobbat. Az eljárás futásideje $\mathcal{O}(n^3)$, várható rátája pedig $0{,}66768 - \varepsilon$ bármely $\varepsilon > 0$ konstansra, amely nagyobb, mint Feo és Khellaf (1990) vagy Hassin és szerzőtársai (1997) determinisztikus algoritmusának 2/3-os approximációs rátája.

5.1.4. Minimális összegű p-klaszterezés

Legyen $G = (V, E)$ irányítatlan, teljes gráf, V csúcshalmazzal, valamint E élhalmazzal, amely elemeninek élhosszára teljesül a háromszög-egyenlőtlenség. A *minimális összegű*

[3] Az eredeti cikkben itt is vétettek egy hibát az elemzésben, melyet később javítottak.

p-klaszterezés (min-sum p-clustering) probléma azt követeli meg, hogy particionáljuk a V csúcshalmazt p darab, esetlegesen előre megadott elemszámú részhalmazra úgy, hogy azon élek hosszainak összege, melyeknek mindkét végpontja ugyanabba a részhalmazba esik, minimális legyen. Guttmann-Beck és Hassin (1998) a problémának arra a változatára adott egy 2-approximációs algoritmust, amelyben a klaszterek elemszáma adott. Eredményük ugyanakkor arra az esetre is érvényes, amikor az elemszámok nem, csupán a klaszterek száma van megadva. Utóbbi esetben a hibakorlát ugyanaz, az algoritmus komplexitása viszont nagyobb.

A Guttmann-Beck és Hassin (1998) által tárgyalt *minimális összegű p-klaszterezés* (MCP) probléma során a feladat, hogy adott p darab pozitív, egészértékű szám egy $\{k_i\}_{i=1}^p$ halmazára, amelyre $\sum_{i=1}^p k_i = |V| = n$, találjuk meg a V egy diszjunkt halmazokból álló $\{P_i\}_{i=1}^p$ partícióját, amelyre $|P_i| = k_i \ \forall i \in \{1, \dots, p\}$ és $\sum_{i=1}^p w(P_i)$ minimális.

Az approximációhoz Guttmann-Beck és Hassin (1998) definiálja a *minimális csillag particionálás problémát* (min star partitioning problem, SPP), amelyben a feladat p darab diszjunkt csúcs $\{v_i\}_{i=1}^p \subset V$ halmazának és a V egy diszjunkt halmazokból álló $\{P_i\}_{i=1}^p$ partíciójának a megtalálása, amelyre $|P_i| = k_i, v_i \in P_i \ \forall i \in \{1, \dots, p\}$ és $\sum_{i=1}^p k_i w(v_i, P_i)$ minimális, ahol $w(v_i, P_i) = \sum_{v_j \in P_i, v_i \neq v_j} w(v_i, v_j)$ jelöli a v_i csúcs és a többi P_i-beli csúcs közötti élek súlyainak összegét.

Guttmann-Beck és Hassin (1998) algoritmusa

Input $G = (V, E)$ gráf, és $k_1, \dots, k_p$ konstansok, melyekre $\sum_{i=1}^p k_i = |V|$

(1) **for** $\{v_i\}_{i=1}^p \subset V$ [minden egyes kombinációra]

$A \leftarrow V \backslash \{v_i\}_{i=1}^p$

$P^{\{v_1,\dots,v_p\}} \leftarrow$ minimális súlyú λ-hozzárendelési probléma megoldása

end

(2) $\{v_1^*, \dots, v_p^*\} \leftarrow \underset{\{v_1,\dots,v_p\} \subset V}{\arg\min} \sum_{i=1}^p k_i w\big(v_i, P_i^{\{v_1,\dots,v_p\}}\big)$

5.6. Algoritmus. Guttmann-Beck és Hassin (1998) algoritmusa

Ennek megoldására Guttmann-Beck és Hassin (1998) megkonstruálja a 5.6. Algoritmust, amely polinomiális időben megtalálja az SPP optimális megoldását. Ennek lényege, hogy az (1)-es lépésben az összes lehetséges módon kiválasztanak az n csúcs közül p-t, mindegyikre megoldanak egy minimális súlyú λ-hozzárendelési problémát, majd a (2)-es

lépésben kapott lehetőségek közül kiválasztják a legkisebbet.

A felhasznált *minimális súlyú λ-hozzárendelési probléma* (lásd Tokuyama és Nakano (1991)) egy szállítási feladat, amely formálisan a következőképpen írható fel. Legyen $\{v_i\}_{i=1}^{p}$ és $A = \{a_i\}_{i=1}^{|V|-p} = V \backslash \{v_i\}_{i=1}^{p}$ a V csúcsainak két diszjunkt halmaza, valamint $k_1, \ldots, k_p$ konstansok, melyekre $\sum_{i=1}^{p} k_i = |V|$. A feladat a V halmaz diszjunkt halmazokból álló $P^{\{v_1,\ldots,v_p\}} = \{P_i^{\{v_1,\ldots,v_p\}}\}_{i=1}^{p}$ partíciójának megtalálása, amelyre $P_i = \{v_i\} \cup A_i \ \forall i \in \{1, \ldots, p\}$, ahol $\{A_i\}_{i=1}^{p}$ az A halmaz diszjunkt halmazokból álló partíciója, és $\sum_{i=1}^{p} \sum_{a_j \in A_i} k_i w(v_i, a_j)$ minimális. Ez pedig Tokuyama és Nakano (1991) algoritmusával $\mathcal{O}(n)$ időben megoldható.

Mivel $\mathcal{O}(n^p)$ nagyságrendű $\{v_1, \ldots, v_p\} \subset V$ részhalmaz képezhető, az alábbi eredmény rögtön adódik.

5.1.9. Tétel. *Az 5.6. Algoritmus egy optimális megoldást ad az SPP problémára. Az algoritmus futásideje* $\mathcal{O}(n^{p+1})$.

Emellett Guttmann-Beck és Hassin (1998) a következő approximációs eredményt is kimondja.

5.1.10. Tétel. *Legyen* $P_1, \ldots, P_p$ *az 5.6. Algoritmus által meghatározott partíció. Ekkor*

$$w(Apx) = \sum_{i=1}^{p} w(P_i) \leq 2w(Opt).$$

Végül azt is belátják, hogy 2 egyenlő méretű halmaz megkövetelése esetén az approximációs ráta nem nagyobb 1,7-nél.

5.2. Kúp optimalizálás

Az utóbbi években több konvex optimalizálási megközelítés is született az MSSC probléma relaxált változatának megoldására (lásd például Peng és Wei (2007) és Awasthi és szerzőtársai (2015)). Ezek az ún. 'kúp programok' polinomiális időben megoldhatóak, könnyebben kezelhetőek, mint az alapprobléma, és egy korlátot is meghatároznak egy adott megoldás szuboptimalitására. A relaxált probléma megoldása alapesetben az eredeti probléma szempontjából nem megengedett megoldás. Ugyanakkor ezek helyreállításával, vagy másképpen "kerekítésével" előállítható olyan megoldás, amely már az eredeti probléma szerint is megengedett. (Rujeerapaiboon és szerzőtársai, 2019)

Rujeerapaiboon és szerzőtársai (2019) elsőként alkalmazza ezt a megközelítést az általuk *elemszámkorlátozott K-közép klaszterezési problémának* (cardinality-constrained K-means clustering problem) nevezett feladatra, amely az egyenletes MSSC feladat általános megfogalmazása tetszőleges elemszámkorlátokkal. Kúp relaxációkat és kerekítő heurisztikákat kombinálnak a probléma megoldására, illetve egy a-posteriori garanciát is megadnak a megoldás optimalitására. Eredményeiket tanulmányuk alapján a következőekben mutatjuk be.

Jelölje $\mathbb{S}^N$ a szimmetrikus, $N \times N$-es (négyzetes) mátrixok halmazát. Jelölje $A, B \in \mathbb{S}^N$ mátrixokra az $A \succeq B$ reláció azt, hogy $A - B$ pozitív szemidefinit, valamint $A \geq B$ azt, hogy $A - B$ elemenként nemnegatív. Legyen továbbá $\langle A, B \rangle = \mathrm{Tr}(AB)$, vagyis az AB mátrixszorzat főátlójában lévő elemeinek összege.

Rujeerapaiboon és szerzőtársai (2019) felírása szerint legyen adva $\xi_1, \ldots, \xi_N \in \mathbb{R}^d$, N darab adatpont. Ezeket $I_1, \ldots, I_K$, K darab klaszterre kell particionálni, rendre $n_1, \ldots, n_K$, $n_1 + \cdots + n_K = N$ klaszterméretekkel úgy, hogy a klasztereken belüli távolságok négyzeteinek összege minimális legyen. Formálisan a feladat a következőképpen adható meg:

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K} \sum_{i \in I_k} \left\| \xi_i - \frac{1}{n_k} \left(\sum_{j \in I_k} \xi_j \right) \right\|^2 \\ \text{s.t.} & (I_1, \ldots, I_K) \in \mathfrak{P}(n_1, \ldots, n_K), \end{array} \tag{5.1}$$

ahol

$$\mathfrak{P}(n_1, \ldots, n_K) = \left\{ (I_1, \ldots, I_K) : |I_k| = n_k \ \forall k, \cup_{k=1}^{K} I_k = \{1, \ldots, N\}, I_k \cap I_l = \emptyset \ \forall k \neq l \right\}$$

a probléma felírása szerinti megengedett partíciók halmaza.[4]

Rujeerapaiboon és szerzőtársai (2019) az (5.1) feladatot először egy ekvivalens kvadratikus hozzárendelési problémává[5] alakítja, majd ezt felhasználva adja meg az 5.2.1. Állításban szereplő MILP (kevert egészértékű lineáris program, mixed-integer linear program) feladatot. Az általános kvadratikus hozzárendelési problémák N dolgozó és N munka esetén N^2 bináris változóval rendelkező MILP alakban írhatóak fel. Ugyanakkor jelen esetben az adatpontok klaszteren belüli permutációs szimmetriájából következően $NK \ll \Omega(N^2)$ bináris változó is elegendő.

[4] A probléma felírásánál az 's.t.' a 'subject to' rövidítése, ami azt jelenti, hogy a megadott célfüggvényt a felsorolt feltételek teljesülése mellett kell optimalizálni.

[5] Kvadratikus hozzárendelési problémáról lásd például Burkard (2013).

5.2.1. Állítás. MILP átírás

Az (5.1) *által megadott klaszterezési feladat ekvivalens MILP átírása az alábbi:*

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\sum_{k=1}^{K}\tfrac{1}{n_k}\sum_{i,j=1}^{N} d_{ij}\eta_{ij}^k \\ \text{s.t.} \quad & \pi_i^k \in \{0,1\}, \eta_{ij}^k \in \mathbb{R}_+ \quad i,j=1,\dots,N, k=1,\dots,K, \\ & \textstyle\sum_{i=1}^{N}\pi_i^k = n_k \qquad\qquad k=1,\dots,K, \\ & \textstyle\sum_{k=1}^{K}\pi_i^k = 1 \qquad\qquad i=1,\dots,N, \\ & \eta_{ij}^k \geq \pi_i^k + \pi_j^k - 1 \quad i,j=1,\dots,N, k=1,\dots,K, \end{aligned} \tag{$\mathcal{P}$}$$

A π_i^k bináris változóra $\pi_i^k = 1$, ha $i \in I_k$, és $\pi_i^k = 0$ egyébként. Optimumban pedig $\eta_{ij}^k = \max\{\pi_i^k + \pi_j^k - 1, 0\} = 1$, ha $i,j \in I_k$ (vagyis $\pi_i^k = \pi_j^k = 1$), és 0 egyébként.

Rujeerapaiboon és szerzőtársai (2019) ezt követően relaxálja a nehezen kezelhető $\mathcal{P}$ MILP felírást egy olyan kúp programozási feladatra, amely majd hatékonyan (vagyis polinomiális futásidőben) számítható alsó és felső korlátokat eredményez a $\mathcal{P}$ problémára. Ehhez az $x_i^k \leftarrow 2\pi_i^k - 1$ változó transzformációt alkalmazzák, így x_i^k értéke $+1$, ha az i-edik adatpontot a k klaszterhez rendeljük, és -1 egyébként. Így a $\mathcal{P}$ MILP egy ekvivalens alakjára a következő SDP (*szemidefinit program*) relaxáció adható meg:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{8}\left\langle D, \textstyle\sum_{k=1}^{K}\tfrac{1}{n_k}\left(M^k + \mathbb{1}\mathbb{1}^T + x^k\mathbb{1}^T + \mathbb{1}(x^k)^T\right)\right\rangle \\ \text{s.t.} \quad & (x^k, M^k) \in \mathcal{C}_{SDP}(n_k) \quad k=1,\dots,K, \\ & \textstyle\sum_{k=1}^{K} x^k = (2-K)\mathbb{1}, \end{aligned} \tag{$\mathcal{R}_{SDP}$}$$

ahol bármely $n \in \mathbb{N}$ természetes számra a $\mathcal{C}_{SDP}(n) \subset \mathbb{R}^N \times \mathbb{S}^N$ halmazt a következőképpen definiáljuk:

$$\mathcal{C}_{SDP}(n) = \left\{ (x,M) \in \mathbb{R}^N \times \mathbb{S}^N : \begin{array}{l} \mathbb{1}^T x = 2n - N, M\mathbb{1} = (2n-N)x \\ \operatorname{diag}(M) = \mathbb{1}, M \succeq xx^T \\ M + \mathbb{1}\mathbb{1}^T + x\mathbb{1}^T + \mathbb{1}x^T \geq 0 \\ M + \mathbb{1}\mathbb{1}^T - x\mathbb{1}^T - \mathbb{1}x^T \geq 0 \\ M - \mathbb{1}\mathbb{1}^T + x\mathbb{1}^T - \mathbb{1}x^T \leq 0 \\ M - \mathbb{1}\mathbb{1}^T - x\mathbb{1}^T + \mathbb{1}x^T \leq 0 \end{array} \right\}.$$

A fenti SDP további relaxációjával felírhatunk egy LP (*lineáris program*) feladatot is. Legyen $\mathcal{R}_{LP}$ az a feladat, amelyet úgy kapunk, hogy az $\mathcal{R}_{SDP}$ felírásban az $(x^k, M^k) \in \mathcal{C}_{SDP}(n_k)$ feltételt lecseréljük az $(x^k, M^k) \in \mathcal{C}_{LP}(n_k)$ tartalmazásra, ahol bármely $n \in \mathbb{N}$ esetén a $\mathcal{C}_{LP}(n)$ politópot $\mathcal{C}_{SDP}(n)$-ből az $M \succeq xx^T$ nemlineáris feltétel eltávolításával kapjuk. Az $\mathcal{R}_{SDP}$ és $\mathcal{R}_{LP}$ relaxációkra Rujeerapaiboon és szerzőtársai (2019) a következő

tételt látja be.

5.2.2. Tétel.

$$\min \mathcal{R}_{LP} \leq \min \mathcal{R}_{SDP} \leq \min \mathcal{P}.$$

Ez természetesen azt jelenti, hogy az $\mathcal{R}_{SDP}$ és $\mathcal{R}_{LP}$ feladatok megoldásai alsó korlátot adnak az eredeti $\mathcal{P}$ probléma optimumára. Emellett Rujeerapaiboon és szerzőtársai (2019) megad egy olyan kerekítő algoritmust, amellyel az $\mathcal{R}_{SDP}$ és $\mathcal{R}_{LP}$ relaxált problémák optimális megoldását helyreállítja abban az értelemben, hogy az a $\mathcal{P}$ egy megengedett megoldását adja (ezt a kerekítő eljárást itt nem közöljük). Ez pedig természetesen a $\mathcal{P}$ optimumának egy felső korlátját határozza meg.

Az $\mathcal{R}_{SDP}$ és $\mathcal{R}_{LP}$ feladatok megoldásának számítási igénye K-ban növekszik. Ugyanakkor, hogyha minden klaszter mérete azonos, vagyis $n_k = n\ \forall k$, akkor $\mathcal{R}_{SDP}$ lecserélhető az egyszerűbb

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{8n} \left\langle D, M^1 + \mathbb{1}\mathbb{1}^T + x^1\mathbb{1}^T + \mathbb{1}(x^1)^T + (K-1)(M + \mathbb{1}\mathbb{1}^T + x\mathbb{1}^T + \mathbb{1}x^T) \right\rangle \\ \text{s.t.} \quad & (x^1, M^1), (x, M) \in \mathcal{C}_{SDP}(n), \quad x^1 + (K-1)x = (2-K)\mathbb{1}, \quad x_1^1 = 1, \end{aligned} \tag{$\mathcal{R}^b_{SDP}$}$$

problémára, amelynek mérete már nem skálázódik K-ban. Hasonlóan, $\mathcal{R}_{LP}$ egyszerűsíthető az $\mathcal{R}^b_{LP}$ problémára, hogyha $\mathcal{R}^b_{SDP}$-ben lecseréljük $\mathcal{C}_{SDP}(n)$-et $\mathcal{C}_{LP}(n)$-re. Az egyenletes klaszterezés relaxációira rögtön következik az alábbi eredmény.

5.2.3. Következmény. $\min \mathcal{R}^b_{LP} \leq \min \mathcal{R}^b_{SDP} \leq \min \mathcal{P}$.

Az $\mathcal{R}^b_{SDP}$ és $\mathcal{R}^b_{LP}$ optimális megoldásainak helyreállítását a 5.7. Algoritmus írja le. A megfogalmazott relaxációk csupán az első klaszter megbízható helyreállítását teszik lehetővé, így az összes klaszter helyreállításához $K-1$-szer kell megoldanunk az $\mathcal{R}^b_{SDP}$ és $\mathcal{R}^b_{LP}$ problémákat, mindig a még hátralévő adatpontokra.

Kerekítő algoritmus az egyenletes klaszterezésre (Rujeerapaiboon és szerzőtársai, 2019)

Input $\mathcal{I}_1 = \{1, \ldots, N\}$ (indexek), $n \in \mathbb{N}$ (klaszterméret), $K = N/n$

Output $I_1, \ldots, I_K$ (indexhalmazok)

(1) **for** $k = 1, \ldots, K-1$

(a) Az optimális $x^1 \in \mathbb{R}^{|\mathcal{I}_k|}$ meghatározása az $\mathcal{R}^b_{SDP}$ vagy $\mathcal{R}^b_{LP}$ feladat megoldásából a $\xi_i, i \in \mathcal{I}_k$ pontokra

(b) Egy $\rho : \{1, \dots, |\mathcal{I}_k|\} \to \mathcal{I}_k$ bijekció megadása, amelyre $x^1_{\rho(1)} \geq \dots \geq x^1_{\rho(|\mathcal{I}_k|)}$

(c) $I_k \leftarrow \{\rho(1), \dots, \rho(n)\}, \quad \mathcal{I}_{k+1} \leftarrow \mathcal{I}_k \backslash I_k$

end

(2) $I_K \leftarrow \mathcal{I}_K$

5.7. Algoritmus. Kerekítő algoritmus az egyenletes klaszterezésre (Rujeerapaiboon és szerzőtársai, 2019)

Ez a helyreállító eljárás determinisztikus, továbbá model- és dimenziófüggetlen. Ezenkívül Rujeerapaiboon és szerzőtársai (2019) belátja, hogy ha az alábbi, szeparációs feltétel teljesül, akkor az általuk megadott relaxációk és az eredeti probléma optimális értékei megegyeznek, és az 5.7. Algoritmus megtalálja az optimális klaszterezést.

(S) *Tökéletes felosztás* (Perfect separation): Létezik az $\{1, \dots, N\}$ egy egyenletes $(J_1, \dots, J_K)$ partíciója, ahol minden $k = 1, \dots, K$ klaszter elemszáma azonosan $|J_k| = N/K \in \mathbb{N}$ és

$$\max_{1 \leq k \leq K} \max_{i,j \in J_k} d_{ij} < \min_{1 \leq k_1 < k_2 \leq K} \min_{\substack{i \in J_{k_1}, \\ j \in J_{k_2}}} d_{ij}.$$

Az **(S)** feltétel azt fogalmazza meg, hogy létezik az adatoknak egy természetesen adódó $(J_1, \dots, J_K)$ klaszterezése, és a legnagyobb klaszter átmérője (bal oldal) kisebb, mint bármely két különböző klaszter közötti legkisebb távolság (jobb oldal).

5.2.4. Tétel. *Ha az* **(S)** *feltétel teljesül, akkor $\mathcal{R}^b_{LP}$ és $\mathcal{P}$ optimális értékei megegyeznek. Továbbá a $(J_1, \dots, J_K)$ klaszterezés optimális $\mathcal{P}$-ben, és az 5.7. Algoritmus előállítja azt.*

Ez azt jelenti, hogy az **(S)** feltétel teljesülése a-posteriori ellenőrizhető, hogyha az alsó korlát megegyezik az optimummal, de természetesen utóbbiból nem feltétlenül következik az előbbi. Rujeerapaiboon és szerzőtársai (2019) megjegyzi, hogy az **(S)** feltétel teljesülése mellett egyszerűbb helyreállító mechanizmusokat is meg lehetne fogalmazni, ugyanakkor az nem valószínű, hogy azok akkor is jól teljesítenének, amikor a feltétel mégsem áll fenn. A szerzők az általuk elvégzett numerikus kísérletek során azt találták, hogy a megadott kerekítő eljárások viszont még akkor is jól teljesítenek, amikor az **(S)** feltétel sérül.

További eredményként a szerzők azt is megmutatják, hogy az alsó korlátok és a kerekítő algoritmus kiterjeszthető az (5.1)-es probléma olyan eseteire is, amelyekben kilógó adatpontok (outlier) is találhatóak, és az ezen pontok által meghatározott klaszter számossága nem ismert.

Rujeerapaiboon és szerzőtársai (2019) egy numerikus elemzést is elvégez, amelyhez 7 darab, a való életből származó adathalmazt használnak fel, köztük az 'Iris' és 'Seeds' egyenlő elemszámú klaszterekkel rendelkező adatokat.[6] Az adathalmazokban a pontok száma 150-210-ig terjed, a dimenziók száma 4-147, a klaszterek száma pedig 2-9. Az elemzés során az $\mathcal{R}_{SDP}$, illetve $\mathcal{R}^b_{SDP}$ relaxációkkal kapott alsó korlátok és a kerekítő eljárással kapott megengedett megoldások vagy egybeestek (ekkor optimumról beszélünk), vagy nagyon közel voltak egymáshoz. Ugyanakkor volt egy eset is, amikor az algoritmus 3 óra alatt nem terminált.

Emellett a tekintett egyenletes klaszterezési problémák esetében az eredményeket összehasonlították az egyenletes klaszterezés problémára alkalmazható heurisztikus eljárásokkal is, többek között Malinen és Fränti (2014) és Costa és szerzőtársai (2017) algoritmusaival (utóbbi eljárásokról lásd a 6.2.5. és 6.2.6. fejezeteket). Elmondható, hogy a heurisztikus eljárások szintén megtalálták az optimumot, és futásidejük nagyságrendekkel kisebb, mint a kúp optimalizálási megközelítésé. Így bár utóbbi egyértelmű garanciát ad az optimalitásra, a gyakorlatban nagy adathalmazok esetén valószínűleg nem praktikus.

[6] Az adatok forrása a UCI Machine Learning Repository, és a `http://archive.ics.uci.edu/ml/` címen érhetőek el.

6. fejezet

Klaszterező eljárások

Általános minimalizálási klaszterezési problémák megoldására valószínűleg az egyik legismertebb módszertan a *klaszterelemzés*. Ez egy ún. *nem felügyelt tanulási módszer* (unsupervised learning method), amely a statisztikai adatelemzés egyik leggyakrabban használt eszköze (Bertoni és szerzőtársai, 2012). Lényege, hogy előzetes információk nélkül feltérképezi az adatok közötti összefüggéseket, és ez alapján olyan csoportokat alakít ki, ahol az egyes klasztereken belüli elemek bizonyos tulajdonságok szerint nagyobb hasonlóságot mutatnak más csoportok elemeihez képest (Ganganath és szerzőtársai, 2014).

A valóélet-beli problémáknál ugyanakkor gyakran állnak rendelkezésünkre plusz információk a klaszterekről, amelyeket érdemes figyelembe venni a klaszterezési folyamat során, mert javíthatja annak teljesítményét (Bertoni és szerzőtársai, 2012). Emellett előfordulnak olyan esetek is, amikor az alkalmazásból adódóan olyan rögzített elemeszám korlátok vannak, amelyeknek a kialakított csoportoknak eleget kell tenniük.

Azokat a problémákat, amelyek valamilyen háttérinformációt is tartalmaznak, *korlátozott klaszterezési problémáknak* (constrained clustering problem) nevezzük, és két csoportra oszthatjuk őket. Egyfelől vannak olyan klaszterezési feladatok, amelyekben a csoportosítandó elemekre előírt, ún. *elem-alapú* (instance-based) korlátok vannak. Ilyen például, amikor két elemnek mindenképpen ugyanabba a csoportba kell kerülnie, vagy éppen ellenkezőleg, két elem mindenképpen különböző csoportban kell, hogy szerepeljen (lásd Wagstaff és Cardie (2000)).

Másfelől olyan klaszterezési problémákkal találkozhatunk, amelyek a csoportok elemszámára előírt, ún. *csoport-alapú* (group-based) korlátokat tartalmaznak (lásd például Bradley és szerzőtársai (2000) vagy Tung és szerzőtársai (2001)). (Bertoni és szerzőtársai,

2012)

Az m-szobatárs probléma az utóbbi kategóriába tartozik. A megoldás keresésére két megközelítést alkalmazunk. Az első során a klaszterelemzséhez kapcsolódó módszereket tekintünk, amelyeknél a megoldásra nem írhatóak elő a csoportokra vonatkozó elemszám korlátok. Hogy ezek esetén is garantálni tudjuk az egyenlő méretű csoportokat, ezeket általunk konstruált, különböző kiegyenlítő eljárásokkal kombináljuk.

A második megközelítés során olyan algoritmusokat alkalmazunk, amelyek a konstrukciójukból adódóan olyan megoldást garantálnak, amelyek eleget tesznek az előre rögztett elemszám korlátoknak. Ez némely algoritmusnál az azonos méretű klasztereknél általánosabb is lehet, és az egyenletes klaszterezés a feladat egy speciális eseteként adódik.

A fejezetben sorra vesszük a 7. Fejezetben elvégzett elemzésben szereplő klaszterező eljárásokat, mindegyiknél külön gondot fordítva az adott módszer hátterére. A klaszterelemzésnél tárgyalt eljárásokhoz a 6.1.4. Fejezetben mutatjuk be az általunk megadott heurisztikákat az alapmódszerek megoldásainak kiegyenlítésére, amelyeket majd a 7.1. Fejezetben értékelünk ki. Emellett Weitz és Lakshminarayanan (1996) páros cseréken alapuló módszerét a 6.2.3. Fejezetben kiterjesztjük hármas cserékre.

6.1. Klaszterelemzéshez kapcsolódó módszerek

A klaszeterelemzés többféle módszer és eljárás összefoglaló neve, melyek célja, hogy előre nem ismert besorolás esetén feltárjuk és felírjuk az egymáshoz leginkább hasonló egyedek csoportjait (Dr. Kovács és szerzőtársai, 2011). Ezek általában véve egyszerűek és gyorsak, ugyanakkor nem tudjuk megmondani, hogy az eredményül kapott megoldás mennyire közelíti jól az optimumot.

6.1.1. Összevonó hierarchikus klaszterezés (`cluster`)

Dr. Kovács és szerzőtársai (2011) alapján bemutatjuk a klaszterező eljárások egyik fő csoportját, a *hierarchikus osztályozást*. Az ebbe a csoportba tartozó módszerek legfőbb sajátossága, hogy nem kell előre megadni a feltételezett vagy a mintában meglévő csoportok számát. Ez a módszer kétféleképpen végezhető. Az *összevonó* hierarchikus eljárás kezdetben minden elemet külön osztálynak tekint, majd lépésenként egy-egy összevonást végez, így végül $n-1$ lépés során az összes egyedet egyesíti. A *felosztó* hierarchikus eljá-

rás egy nagy csoportból indul ki, majd valamilyen döntési kritérium alapján kettéosztja a megfigyeléseket, így az eljárás $2^{n-1}-1$ felosztás megvizsgálása után fejeződik be. Mivel a gyakorlatban általában az n nagy, így a magas lépésszám miatt nem alkalmazzák.

A hierarchikus klaszterezés módszerei egy távolságmérték és egy összevonó eljárás kombinációjaként állnak elő. A választható távolságmértékek például a négyzetes euklideszi távolság, a Csebisev metrika, a Manhattan metrika, stb. Az összevonó eljárás lehet 'egyszerű lánc', 'teljes lánc', 'átlagos lánc', 'centroid', 'medián' vagy 'Ward'. Az egyszerű lánc és medián módszerek jellemzője a lánchatás, vagyis bizonyos elemeket közbeeső elemek láncolata révén kapcsolnak össze. A teljes lánc, átlagos lánc vagy centroid módszerekkel végzett osztályozás esetén, egy-egy klaszter elemei egymáshoz nagyon közeliek lesznek. A teljes lánc módszer egyenlő átmérűjő, míg a Ward módszer egyenlő elemszámú klaszterek kialakítására törekszik.

Az összevonási folyamat ábrázolása *dendrogramon* történik. Ez egy speciális kétdimenziós ábra, melynek egyik tengelyén az összevont elemeket látjuk, a másik tengelyén pedig (25 maximális távolságértékre átskálázva) azt a távolságértéket, amelynél az összevonás megtörtént. Kezdetben, vagyis a 0 távolsági szinten minden elem egy-egy csoportot alkot, míg a végén, a 25-ös távolsági szinten már minden elem összevonásra került egy nagy csoportba. Az eljárás miként jéből fakadóan egy alkalmas 'vágással' ki tudunk választani egy olyan állapotot, amelynél pont k klaszter volt, ugyanakkor nem garantált, hogy ezen klaszterek elemeinek száma megyegyezik.

A 7. Fejezetben elvégzett elemzésbe bevonunk egy hierarchikus módszert is egy alkalmas vágással, amely biztosítja a csoportok megfelelő számát. Ez távolságmértékként négyzetes euklideszi távolságot, összevonó eljárásként pedig Ward módszert alkalmaz, a kísérletek során pedig `cluster` névvel hivatkozunk rá.

6.1.2. k-közép++ (`kmeans`)

Dr. Kovács és szerzőtársai (2011) alapján a másik fő típus a *nemhierarchikus osztályozás*. Az ebbe a csoportba tartozó módszerek előre meghatározott számú csoportra bontják a mintát, az eljárások általános menete pedig a következő:

1. Kezdő klaszterközéppontok meghatározása.
2. A pontok besorolása a klaszterekbe.
3. Új klaszterközéppontok számítása.

4. Az előző két pont ismétlése, amíg már nem történik lényeges változás.

Ide tartozik a *k-közép* (k-means) algoritmus is, amely az MSSC optimalizálási probléma megoldására alkalmazott legnépszerűbb eljárás (Costa és szerzőtársai, 2017). A "k-közép" kifejezést először MacQueen (1967) használta, de az algoritmus alapötlete Steinhaus (1956) nevéhez kötődik (Ganganath és szerzőtársai, 2014). A sztenderd k-közép algoritmust 1957-ben Lloyd (1982) javasolja (ezért Lloyd algoritmusaként is hivatkoznak rá), de csak később került publikálásra.

Az MSSC probléma számos tulajdonsága közül Costa és szerzőtársai (2017) a következő hármat emelik ki:

1. A klaszteren belüli varianciát minimalizálja, míg a klaszterek közötti távolságokat maximalizálja. (Späth, 1980)
2. Ha a klaszterközéppontok adottak, akkor a lokális optimalitás következtében minden egyes adatpontot a hozzá legközelebbi középponthoz sorolunk.
3. Ha adottak a hozzárendelések, akkor az elsőrendű optimalitási feltételek következtében a klaszterközéppont megegyezik a klaszterhez rendelt adatpontok alapján meghatározott centroiddal. Következésképpen, az MSSC-re úgy is tekinthetünk, mint egy kombinatiorikai optimalizálási problémára. Ennek megfelelően a kérdést úgy is fel lehet tenni, hogy hogyan csoportosítsuk az elemeket, a klaszterközéppontok pedig a csoportosításnak megfelelően egyértelműen adódnak.

A k-közép eljárás a 2. és 3. pontban leírt tulajdonságokban rejlő lehetőségeket aknázza ki. Első lépésként véletlenszerűen kiválasztja a kezdő klaszterközéppontokat. Ezután az összes többi pontot a hozzá legközelebbi klaszterközépponthoz sorolja, amellyel meghatározza a kezdő klasztereket, majd klaszterenként újraszámolja a középpontokat. Az utóbbi két lépést egészen addig ismétli, amíg a folyamat nem stabilizálódik. A távolságok összege minden lépésben monoton csökken, így a folyamat alatt egyik csoportosítás sem ismétlődik, és mivel csak véges sok állapot lehet, így az algoritmus biztosan terminál. Habár nem lehet vizsgálni, hogy az eredmény mennyire közelíti az optimumot, a gyakorlatban egyszerű és gyors módszert kínál egy megengedett megoldás megkonstruálására.

Arthur és Vassilvitskii (2007) úgy módosították az előző algoritmust, hogy a klaszterközéppontok meghatározásához az első lépésben véletlenszerűen kiválasztanak az elemek közül egy c_1 pontot, majd a $c_i, i = 2, \ldots, k$ klaszterközepeket úgy adják meg, hogy az összes pont X halmazából egy x választásának valószínűsége $\frac{D(x)}{\sum_{x \in X} D(x)}$, ahol $D(x)$ jelöli

az x-nek és a hozzá legközelebbi klaszterközéppontnak a távolságát. Vegyük észre, hogy $D(x)$ az $x = c_1, \ldots, c_{i-1}$ pontokra 0. Ezt követően a két algoritmus lépései megegyeznek. Ezzel a kis változtatással a tesztelések alapján az algoritmus mind gyorsaság, mind pontosság tekintetében jobban teljesített. A MATLAB-ban található `kmeans` program is ezt az algoritmust használja, a kísérletek során pedig mi is ezt alkalmaztuk.

6.1.3. fuzzy c-közép egyenlő klaszterméretekkel (`eqFCM`)

A *fuzzy c-közép* (fuzzy c-means, FCM) eljárás általános működését tekintve a k-közép algoritmusra hasonlít, ugyanis alapvetően ez is a klaszterpontok meghatározásának és a hozzárendelések újraszámításának lépéseit alternálja. A 'fuzzy' jelző, melynek jelentése elmosódott vagy életlen vonalú, jól kifejezi a módszer lényegét. Az eljárás eredményeképpen nem egy bináris, egyértelmű hovatartozást kapunk, hanem azt, hogy mely adat mekkora mértékben tartozik egy-egy csoporthoz. Ez a mérték egy konstans, 0 és 1 közötti szám, amely így egyféle súlyozásnak is tekinthető. Höppner és Klawonn (2008) a klaszterezési feladat megoldásra javasolt algoritmust a következőképpen írja fel.

A *fuzzy c-közép* klaszterező eljárás az $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ pontokat c klaszterre particionálja. Az i-edik ($1 \leq i \leq c$) klasztert egy $p_i \in \mathbb{R}^d$ ún. *prototípus* (prototype) reprezentálja, továbbá azt, hogy az x_j pont mekkora mértékben tartozik a p_i prototípushoz, vagy másképpen az i-edik klaszterbe, egy $u_{ij} \in [0, 1]$ *hozzátartozási mérték* (membership degree) mutatja. Az u_{ij}-k együtt egy $U \in \mathbb{R}^{c \times n}$ hozzátartozási mátrixot adnak, emellett a következőt követeljük meg rájuk:

$$\forall\ 1 \leq j \leq n: \quad \sum_{i=1}^{c} u_{ij} = 1. \tag{6.1}$$

A klaszterezési eljárás a

$$J_m = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^m d_{ij}, \qquad d_{ij} = \|x_j - p_i\|^2 \tag{6.2}$$

célfüggvény minimalizálásával történik a (6.1) feltétel mellett. Ha az euklideszi távolság az x_j pont és a p_i prototípus között nagy, akkor J_m minimalizálása esetén az u_{ij} értéke várhatóan kicsi, 0-hoz közeli lesz, míg ha x_j és p_i közel vannak egymáshoz, a hozzátartozási mérték magas lesz. A J_m minimalizálása hatékonyan elvégezhető váltakozó optimalizáció mellett. Először a prototípusok szerint, feltételezve, hogy az u_{ij}-k konstansok, majd pedig a hozzátartozási mértékek szerint, konstans prototípusok feltételezése mellet minimalizál-

juk (6.2)-t. Így mind a prototípusokra, mind a hozzátartozási mértékekre zárt képletet kapunk:

$$\forall\ 1 \leq i \leq c: \quad p_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{6.3}$$

$$u_{ij} = \begin{cases} \frac{1}{\sum_{l=1}^{c} \left(\frac{\|x_j - p_i\|^2}{\|x_j - p_l\|^2} \right)^{\frac{1}{m-1}}} & \text{, ha } I_j = \emptyset \\ \frac{1}{|I_j|} & \text{, ha } I_j \neq \emptyset, i \in I_j \\ 0 & \text{, ha } I_j \neq \emptyset, i \notin I_j, \end{cases} \tag{6.4}$$

ahol $I_j = \{k \in \mathbb{N}_{\leq c} \mid x_j = p_k\}$.

Legyen $m > 1$ (általában $m = 2$), és jelölje ΔJ_m a célfüggvény értékének változását az egyes iterációk között. Jelölje továbbá $\varepsilon > 0$ azt a küszöböt, amelynél ha kevesebbet javít az algoritmus a célfüggvény értékén, vagyis amennyiben $\Delta J_m < \varepsilon$, akkor leáll. Ekkor a klaszterező eljárás pszeudokódját a 6.1. Algoritmus írja le.

Fuzzy c-közép eljárás (Fuzzy c-means, FCM, Höppner és Klawonn (2008))

---

(1) Kezdő p_i prototípusok meghatározása (véletlenszerűen)

(2) **while** ($\Delta J_m \geq \varepsilon$)

  (a) $u_{ij} \leftarrow$ hozzátartozási mértékek újraszámítása (6.4) alapján

  (b) $p_i \leftarrow$ prototípusok újraszámítása (6.3) alapján

  end

6.1. Algoritmus. Fuzzy c-közép eljárás (Fuzzy c-means, FCM, Höppner és Klawonn (2008))

Höppner és Klawonn (2008) úgy módosították az előző algoritmust, hogy a (6.1) és (6.2) egyenletekhez hozzávették a

$$\sum_{j=1}^{n} u_{ij} = \frac{n}{c} \tag{6.5}$$

korlátot is, ezzel megfogalmazva a probléma *egyenlő klaszterméreteket* (equi-sized clusters) megkövetelő verzióját, amelyben a klaszterek azonos mértékben fednek le pontokat.

Levezették, hogy az m-et 2-nek választva a következőképpen határozhatjuk meg az u_{ij} értékeket:

- megoldjuk $\beta_1, \ldots, \beta_c$-re az alábbi egyenletrendszert:

$$-\sum_{j=1}^{n} \frac{\sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{2\sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} + \beta_k \sum_{j=1}^{n} \frac{1}{2d_{kj}} = \frac{n}{c} - \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} \qquad k = 1, \ldots, c, \tag{6.6}$$

- kiszámítjuk az $\alpha_1, \ldots, \alpha_n$ értékeket, ahol

$$\alpha_j = \frac{2 - \sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{\sum_{i=1}^{c} \frac{1}{d_{ij}}},$$

- majd az

$$u_{ij} = \frac{\alpha_j + \beta_i}{2d_{ij}}$$

képlet alapján megadjuk a hozzátartozási mértékeket.

A p_i-k kiszámításának módja nem változott, így a 6.1. Algoritmus lényegében annyiban módosul, hogy a (2a) lépésben az u_{ij} értékeket a fenti módon határozzuk meg. Megjegyezzük, hogy Höppner és Klawonn (2008) a módszer tesztelésénél azt tapasztalta, hogy habár a hozzátartozási értékek minden klaszterre $\frac{n}{c}$-t adtak eredményül, kis százalékban előfordultak negatív u_{ij} értékek, így az az interpretáció, miszerint $\sum_{j=1}^{n} u_{ij}$ a klaszter által lefedett pontok száma, nem teljesen korrekt.

Vegyük észre, hogy a (6.6) által felírt egyenletrendszer sorai lineárisan összefüggőek, ezért az implementációban a megoldás keresésénél általánosított inverzet használunk. Vegyük észre továbbá, hogy az algoritmus megoldása nem egyenlő elemszámú csoportokat ad eredményül, hanem olyan csoportokat határoz meg, amelyekre a hozzátartozási mértékek összege egyenlő. Ahhoz, hogy a csoportosítás egyértelmű legyen, minden elemet abba a klaszterbe sorolunk, amelybe a hozzátartozási mérték alapján a legnagyobb mértékben tartozik. Ez a módszer adja az `eqFCM` eljárást.[1]

6.1.4. Kiegyenlítő eljárások `eq1-6`

Az összevonó hierarchikus klaszterezés alkalmas vágással (`cluster`), a k-közép++ algoritmus (`kmeans`) és a fuzzy c-közép egyenlő klaszterméretekkel módszer (`eqFCM`) közül egyik sem ad garantáltan egyenlő elemszámú csoportokat, ezért megadunk hat heurisztikus eljárást a csoportok kiegyenlítésére. Ezek mindegyike a 0. lépésben ellenőrzi, hogy a klaszterek száma k-val egyenlő-e. Amennyiben ez nem teljesül, úgy a legnagyobb elemszámmal

[1] Az 6.1. Algoritmus által megadott FCM eljárás megtalálható MATLAB-ban, így ezt felhasználtuk az implementáció során.

rendelkező klaszterből kiválasztjuk azt az elemet, amely a legtávolabb helyezkedik el a klaszter középpontjától, és ebből egy új, egyelemű csoportot hozunk létre. Ezt addig ismételjük, amíg már nem maradnak üres klaszterek. Ezt követően az egyes eljárások működési elvei az alábbiak szerinti:

1. kiegyenlítő módszer (`eq1`)

   A legnagyobb elemszámú klasztertől kezdve, egymás után egyenlíti ki a csoportokat. Egy adott csoport esetén a többi klaszterközépponttól való távolság alapján sorol át annyi elemet a klaszterből, amennyi az ideális csoport elemszám eléréséhez szükséges. Minden lépésben azt az elemet helyezzük át, amelyik a legközelebb van valamely másik klaszter középpontjához. Az átsorolásokat követően 'letiltjuk' (másképpen megjelöljük) az adott csoportot, hogy ne kerülhessenek vissza az elemek, és átlépünk a következő legnagyobb elemszámú klaszterre.

   Ha a maximálisan szükséges lépések számára szeretnénk meghatározni egy felső becslést, akkor azt az esetet kell tekintenünk, amelyben minden lépésben a lehető legtöbb elemet kell mozgatnunk. Erre akkor kerül sor, amikor egy klasztert leszámítva, minden csoportban csupán egy elem van, és az átsorolásoknál a többlet mindig ugyanabba a másik klaszterbe kerül át.

   Ekkor a legnagyobb klaszter elemeinek száma $n - k + 1$. Ennek a klaszternek a kiegyenlítéséhez így $n - k - m + 1$ átsorolás szükséges. A következő legnagyobb csoport elemszáma $n-k-m+2$, amelynek a kiegyenlítéséhez $n-k-2m+2$ átsorolás szükséges. Hasonló elvet követve a $k-1$-edik legnagyobb csoport kiegyenlítéséhez $m-1$ elemet kell mozgatni. Így bármely kiegyenlítés során legfeljebb $\frac{k(k-1)(m-1)}{2}$ lépésre van szükség. Az eljárás pszeudokódját a 6.2. Algoritmus írja le.[2]

Az 1. kiegyenlítő eljárás (`eq1`)

Input $C = \{C_1, \ldots, C_k\}$ nemüres klaszterek $c = \{c_1, \ldots, c_k\}$ klaszterközéppontokkal, ahol $c_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$ a C_i klaszter középpontja, valamint a klaszterek egyenlő m mérete

[2] A további kiegyenlítő eljárások ennek kisebb módosításaival következnek, így azok pszeudokódjait nem közöljük.

(1) $i \leftarrow \arg\max_i \{|C_i| : C_i \in C\}$ [a legnagyobb klaszter indexe]

(2) **while** $(|C_i| > m)$

$j, l \leftarrow \arg\min_{j,l} \{d(x_j, c_l) : x_j \in C_i, c_l \in c, l \neq i\}$

$C_i \leftarrow C_i \backslash x_j,\ C_l \leftarrow C_l \cup \{x_j\},\ c_l$ frissítése

end

(3) $C\backslash C_i,\ c\backslash c_i,\ i \leftarrow \arg\max_i \{|C_i| : C_i \in C\}$ [halmazok és index frissítése]

(4) **if** $(|C_i| > m)$

goto Step (2)

end

6.2. Algoritmus. Az 1. kiegyenlítő eljárás (`eq1`)

2. kiegyenlítő módszer (`eq2`)

A legnagyobb elemszámú klasztertől kezdve 'lecsorgatja' a többletet, amíg a csoportok ki nem egyenlítődnek. Egy lépésben átsorol egy adott klaszterből egy elemet valamely másik csoportba a klaszterközéppontoktól való távolság alapján. Mindig azt az elemet helyezzük át, amelyik a legközelebb van valamelyik másik csoport középpontjához. Az átsorolás után letiltja azt a klasztert, amelyből áthelyezésre került az elem, majd átlép a következő legnagyobb elemszámú csoportra. Ezt iteráljuk, amíg van olyan még nem letiltott csoport, amelynek elemszáma nagyobb az ideálisnál, egyébként pedig feloldjuk a tiltásokat, és újraindítjuk az algoritmust.

Mivel az optimálisnál kisebb elemszámú csoportokból sohasem veszünk el elemeket, és a tiltások feloldása előtt egy elem biztosan átkerül a többlettel rendelkező csoportokból a hiánnyal rendelkezőkbe, az algoritmus biztosan terminál.

A kiegyenlítéshez szükséges lépésszám felső korlátját úgy kaphatjuk meg, ha a lehető legrosszabb elosztást tekintve nézzük a szükséges lépések számát. Ez akkor fordul elő, amikor $k-1$ csoportban csupán 1 elem szerepel, és az összes többi elem egy klaszterben található. A legrosszabb esetben az elemeket minden hiánnyal rendelkező csoporton keresztül tudjuk csak eljuttatni végső helyükre, így a maximálisan szükséges lépések száma ebben az esetben is $\frac{k(k-1)(m-1)}{2}$.

3. kiegyenlítő módszer (`eq3`)

A legkisebb elemszámú klasztertől kezdve, egymás után egyenlíti ki a csoportokat az 1. kiegyenlítő módszerhez hasonló módon. Egy adott csoportba annyi elemet sorolunk át, amennyi az ideális csoport elemszám eléréséhez szükséges. Az átsorolás a hiánnyal rendelkező csoport klaszterközéppontja és a többi, legalább két elemmel rendelkező csoportban lévő elemek távolsága alapján történik. Mindig azt az elemet helyezzük át, amelyik a legközelebb van az éppen kiegyenlíteni kívánt klaszterhez. Az algoritmus legfeljebb $(k-1)(m-1)$ áthelyezés után biztosan terminál.

4. kiegyenlítő módszer (`eq4`)

   A legkisebb elemszámú klaszter felől indulva, és afelé 'csorgatja' a többletet, amíg a csoportok ki nem egyenlítődnek. Egy lépésben az éppen kiegyenlíteni kívánt klaszter középpontjának és a többi, legalább két elemmel rendelkező csoport elemeinek a távolságát tekinti. Mindig azt az elemet sorolja át, amelyik a legközelebb van az aktuálisan tekintett klaszterhez. Az algoritmus legfeljebb $(k-1)\left(\frac{(m-2)k}{2}+1\right)$ áthelyezés után biztosan terminál.

5. kiegyenlítő módszer (`eq5`)

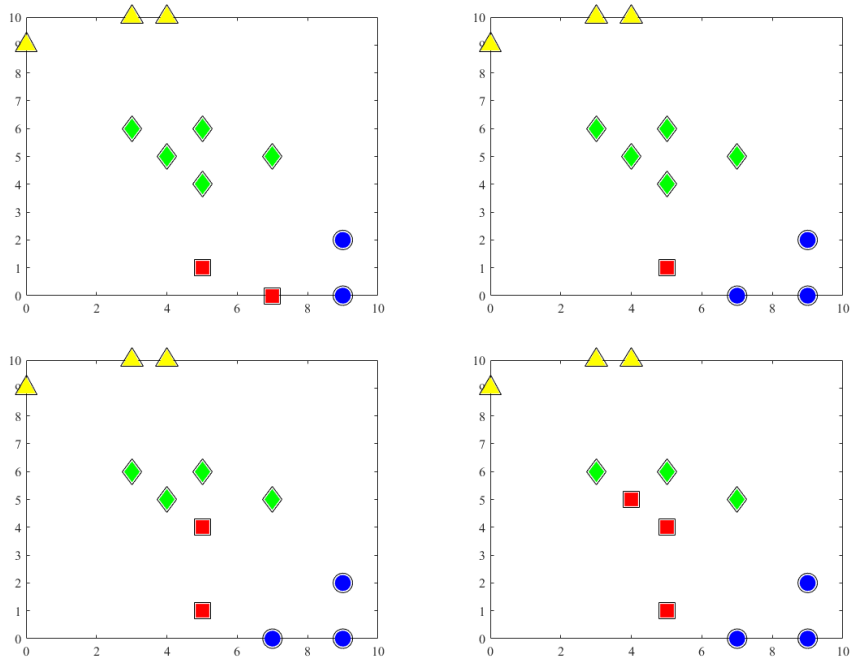
   Az eljárás az 1. és a 3. kiegyenlítő módszereket ötvözi. Miután végrehajtotta a legnagyobb elemszámú csoport kiegyenlítését, letiltja azt, majd a legkisebb klaszter kiegyenlítésével folytatja, amelyet szintén letilt, ha azzal végzett. Ezt a két fázist iterálja egymás után, amíg vannak az ideálistól eltérő méretű klaszterek.

6. kiegyenlítő módszer (`eq6`)

   Egyszerre csökkenti a többletet a legnagyobb elemszámú klaszter felől és tölt fel egy hiányt a legkisebb csoportoknál. Egy lépésben átsorol egy elemet az elérhető (nem letiltott) legnagyobb elemszámú klaszterből, letiltja azt, majd ha még lehetséges, átrak egy elemet a legkisebb csoportba, és ezt a klasztert is letiltja. A két fázist addig iteráljuk, amíg az összes csoport ki nem egyenlítődik.

   Az algoritmus a letiltott csoportok feloldása előtt biztosan feltölt egy hiányt, így az algoritmus véges sok lépés után biztosan terminál.

A 3. módszer egy futását szemlélteti a 6.1. ábra, ahol a bal felső sarokban jelenik meg a kiinduló állapot, és sorfolytonosan következnek az algoritmus egyes lépéseiben kapott beosztások.



6.1. ábra. Az `eq3` kiegyenlítő eljárás egy futását szemléltető ábrák.

A későbbi összehasonlítás során, amikor ötvözzük a kiegyenlítő eljárások valamelyikét egy nem egyenlő csoportokat eredményező heurisztikus módszerrel, akkor azt a kiegyenlítő eljárás nevének utótagként való alkalmazásával tesszük meg. Például az összevonó hierarchikus klaszterezés és az `eq3` kiegyenlítő eljárás alkalmazása révén kapott, biztosan egyenlő számú klasztereket adó heurisztikát `cluster_eq3` névvel jelöljük. Mivel a kiegyenlítő eljárások önmagukban is alkalmazhatóak egyenlő elemszámú klaszterek létrehozására, ezért ezeket így is bevesszük a későbbi összehasonlításba.

6.2. Klaszterező eljárások elemszám megkötésekkel

A következőekben olyan eljárásokat tekintünk, amelyekkel garantálható a megkonstruált csoportosításban a klaszerek egyenlő mérete. Elsőként a fuzzy c-közép egyenlő klaszterméretekkel módszerből vezetünk le egy ilyen eljárást, majd ezt követően az irodalomban található, a feladat megoldására legalkalmasabbnak tűnő heurisztikákat vesszük sorra.

6.2.1. Fuzzy c-közép egyenlő elemszámú klaszterekkel (`eqFCMv2`)

A fuzzy c-közép egyenlő klaszterméretekkel eljárás eredményeképpen egy olyan hozzátartozási mátrixot kapunk eredményül, amely egyértelműen meghatározza, hogy melyik pont

mekkora mértékben tartozik az egyes klaszterekhez. Ez alapján megkonstruálhatunk egy olyan csoportosítást is, amelyben minden klaszterbe azonos számú elemet sorolunk.

Az eljárás során az u_{ij} hozzátartozási mértékek szerint haladunk csökkenő sorrendben. Egy adott u_{ij} változó esetén, ha a j elemet még nem osztottuk be sehová sem, és az i csoportban van még szabad hely, akkor a j elemet az i csoporthoz rendeljük. Ellenkező esetben ugrunk a következő u_{ij} értékre, míg az összes elemet be nem osztottuk. Az így kapott módszerre `eqFCMv2` névvel hivatkozunk a 7. Fejezetben az elemzés során.

6.2.2. Lotfi-Cerveny-Weitz (`LCW`) és egyéb kapcsolódó eljárások

Weitz és Lakshminarayanan (1996, 1998) a *maximálisan diverz* (maximally diverse) hallgatói csoportok kialakításának problémáját tekinti. A problémát kvadratikus egészértékű programozási feladatként írják fel az alábbi formában:

$$\max \sum_{p=1}^{G} \sum_{i=1}^{N-1} \sum_{j>i}^{N} d_{ij} x_{ip} x_{jp}, \tag{6.7}$$

s.t.

$$\sum_{p=1}^{G} x_{ip} = 1, \ i = 1, \ldots, N,$$

$$\sum_{i=1}^{N} x_{ip} = S, \ p = 1, \ldots, G,$$

ahol N jelöli a hallgatók számát, G a csoportok számát, $N/G = S$ pedig a csoportonkénti hallgatók számát. Továbbá

$$d_{ij} = \text{az } i \text{ és } j \text{ hallgatók közötti távolság},$$

$$x_{ip} = 1, \text{ ha az } i \text{ hallgató a } p \text{ csoportban van és}$$

$$0 \text{ egyébként.}$$

Tehát a feladat lényege olyan csoportok kialakítása, amelyekre a csoportokon belüli, elemek közötti távolságok összege (vagyis a csoportokon belüli diverzitás, vagy másképpen különbözőség) az összes csoportra összegezve maximális. Weitz és Lakshminarayanan (1998) a probléma megoldására öt különböző heurisztikus algoritmust tesztelt valós adatokon.

Ezek közül az egyik a 'Weitz-Jelassi' (WJ, Weitz és Jelassi (1992)) módszer. Ez a konstruktív algoritmus úgy működik, hogy először kiválaszt véletlenszerűen egy hallga-

tót, akit besorol az 1. csoportba. Ezután a következő csoportba a még sehová sem sorolt hallgatók közül kiválasztja azt, amelynek a távolsága a legkisebb a legutóbb besorolt hallgatótól, és így tovább. Mivel az algoritmus egyszerű és determinált, a megbízhatóbb eredmény érdekében célszerű N-szer lefuttatni, mindig más kezdőértéket választva. Az algoritmus egy futás esetén $N(N-1)/2$ összehasonlítást végez, így a futásidő összességében $\mathcal{O}(N^3)$.

Két másik módszer az 'Arani-Lotfi' (AL, Arani és Lotfi (1989)) és az 'Arani-Lotfi-Weitz' (ALW) eljárások. Ezek egy tetszőleges kezdeti beosztásból indulnak ki, amelyre teljesül, hogy minden csoportban ugyanannyi hallgató van. Ezután minden csoportból 1-1 hallgatót kiválasztva, az így összesen G elemet optimálisan újraosztják, majd újabb G hallgatót választanak, és így tovább. Az egyedüli különbség a két módszer között abban rejlik, hogy az adott módszer hogyan választ ki minden csoportból 1-1 hallgatót. Az AL először újraoszt G hallgatót, majd másik G hallgatót, stb., egészen addig, amíg minden hallgatót kiválasztott egyszer, majd az így kialakult S csoportra visszafelé haladva is végrehajtja az újrarendezéseket. Egészen addig ismétli az előre-vissza ciklusokat, amíg már nem történik több csere. Ettől eltérően az ALW minden cserénél véletlenszerűen választ minden csoportból 1-1 hallgatót, és azokat osztja újra optimálisan. Itt egy újrarendezés tekinthető úgy is, mint egy maximális súlyú teljes párosítás megtalálása egy páros gráfban. Ez pedig Kuhn (1955) Magyar módszerével a páros gráf csúcsaiban polinomiális, jelen esetben $\mathcal{O}(G^3)$ időben megoldható.

A még nem említett két módszer a 'Lotfi-Cerveny' (LC, Lotfi és Cerveny (1991)) és a 'Lotfi-Cerveny-Weitz' (LCW, Weitz és Lakshminarayanan (1996)) heurisztikák. Ezek is egy tetszőleges kezdő megoldásból indulnak ki, amelyre teljesül, hogy minden csoportban ugyanannyi hallgató van, majd hasonló eljárás során, párok cseréjével javítják a távolságok összegét. Az LC először kiválaszt egy, a csere szempontjából legjobbnak tűnő csoportot, majd azon belül vizsgálja a javítás lehetőségét. Ezzel szemben az LCW az összes másik csoportban lévő hallgatóval történő cserék közül választja a legjobbat. A futási idő tekintetében csupán annyit tudunk mondani, hogy az LC egy-egy iteráció esetén $G + 2S - 2$, míg az LCW $2(N - S) - 1$ összehasonlítást vagy számítást végez.

Meg kell jegyeznünk, hogy egyik cseréket végrehajtó módszer esetén sem lehet megbecsülni az algoritmus terminálásához szükséges iterációk számát. Mindezzel együtt az

eredeti tanulmány szerint a valós adatokon való tesztelés[3] során az LCW nagyon jó eredményeket produkált, valamint konzisztensen ez teljesített a legjobban a vizsgált módszerek közül. Ennélfogva bevesszük a 7. Fejezetben bemutatott elemzésbe, és `LCW` névvel hivatkozunk rá. Az LCW eljárás pszeudokódját a 6.3. Algoritmus írja le.

Lotfi-Cerveny-Weitz (LCW) Algoritmus (Weitz és Lakshminarayanan, 1996)

---

(1) Egy tetszőleges $X = [x_{ip}]$ kezdőmegoldás megkonstruálása, ahol $x_{ip} = 1$, ha $i = (p-1) * S + 1, (p-1) * S + 2, \ldots, (p-1) * S + S$ és $p = 1, 2, \ldots, G$, valamint 0 minden más esetben.

(2) $R \leftarrow DX$, ahol $D = [d_{ij}]$ az elemek közötti távolságok mátrixa.
Flag $\leftarrow$ false [ezzel jelöljük, ha csere történik a későbbiek során]
$i \leftarrow 0$

(3) *Megnézzük, hogy lehet-e javítani úgy, hogy az i elemet kicseréljük egy másik csoportban lévő elemmel.*

$i \leftarrow i + 1$
if $(i \leq N)$
$t \leftarrow$ az a csoport, amelyben az i elem van, vagyis amelyre $x_{it} = 1$
$k \leftarrow \arg\max_{j \in J} w_j$, ahol $w_j = (r_{iq} - r_{it}) + (r_{jt} - r_{jq}) - 2d_{ij}$ és
$J = \{j | x_{jq} = 1, 1 \leq q \leq G, q \neq t\}$
if $(w_k > 0)$
$x_{kq} \leftarrow 0, x_{kt} \leftarrow 1, x_{iq} \leftarrow 1, x_{it} \leftarrow 0$ [kicseréljük a két elemet]
$R \leftarrow DX$ [frissítjük az R mátrixot]
Flag $\leftarrow$ true [jelöljük, hogy csere történt]
end if
goto Step (3)
end if

(4) *Megnézzük, hogy történt-e csere a legutóbbi iteráció során, és ha nem, akkor az algoritmus terminál.*

if (Flag == false)

---

[3]Az adatok forrása a Stern School, NYU, és 360 hallgató adatait tartalmazta. A hallgatókat először 60 fős blokkokba sorolták be, majd mindegyik blokkot 6 fős csoportokra osztották.

stop [nem lehet tovább javítani cserékkel]

else

Flag ← false, $i \leftarrow 0$, **goto** Step (3)

end if

6.3. Algoritmus. Lotfi-Cerveny-Weitz (LCW) Algoritmus (Weitz és Lakshminarayanan, 1996)

6.2.1. Megjegyzés. Vegyük észre, hogy az $R \in \mathbb{R}^{N \times G}$ mátrix r_{ip} eleme az i elem és a p csoport esetén megadja, hogy mennyivel járul hozzá az i elem a teljes differenciához azzal, hogyha a) $i \in p$, vagy hogyha b) $i \notin p$ elem átkerül p-be.

Vegyük észre továbbá, hogy az R mátrix frissítésénél elegendő a t és q csoportoknak megfelelő oszlopokat frissíteni a következő formában:

$$R_{\bullet q} \leftarrow R_{\bullet q} - D_{\bullet k} + D_{\bullet i},$$
$$R_{\bullet t} \leftarrow R_{\bullet t} - D_{\bullet i} + D_{\bullet k},$$

ahol $X_{\bullet j}$ az X mátrix j-edik oszlopát jelöli. Ebből következik, hogy a frissítés $\mathcal{O}(N)$ időben végrehajtható.

6.2.3. Az LCW algoritmus hármas cserékkel (`LCWv2`, `LCWv3` és `LCWv4`)

Az LCW algoritmus akkor terminál, hogyha már nem lehet párok cseréjével javítani a célfüggvényértéken. Ugyanakkor az optimalizálási feladat szempontjából az LCW módszer megoldása nem feltétlenül optimális. Ugyanis előfordulhat, hogy habár egy pár cseréjével már nem lehet javítani a célfüggvényértéken, egy hármas vagy nagyobb méretű cserével - ahol az elemek különböző klaszterekből származnak, és egyik sem marad a helyén - még lenne rá lehetőség. Vagyis az LCW heurisztika során meglévő stabilitási feltétel könnyedén kiterjeszthető úgy, hogy az eredeti algoritmust kiegészítjük hármas, vagy akár nagyobb elemszámú cserékkel is.

Természetesen minél nagyobb elemszámú lehetséges cseréket veszünk be az eljárásba, annál nagyobb az esélye, hogy közelebb kerülünk az optimumhoz. Ugyanakkor sejthető, hogy minél nagyobb potenciális cseréket tekintünk, annál több lépést kell végrehatjania az algoritmusnak, és így a célfüggvény javítása a futásidő rovására mehet.

Könnyen meggondolható, hogy $s = 2$ elem cseréje csak egyféleképpen valósítható meg

úgy, hogy egyik elem se maradjon a helyén. $s = 3$ esetén a lehetséges cserék száma 2, $s = 4$-re ugyanez már 9. A lehetséges cseréket mutatja az előző esetekre a 6.2. ábra. Általános k elem esetén a cserék számát pedig a következő állítás adja meg.

6.2.2. Állítás. *Legyen σ egy n-edfokú permutáció. Azt mondjuk, hogy i a σ fixpontja, ha $\sigma(i) = i$, azaz i helyben marad. Azon n-edfokú permutációk számát, amelyeknek nincs fixpontja, a*

$$D_n = nD_{n-1} + (-1)^n, D_0 = 1$$

rekurzív képlet adja.

A	B
B	A

A	B	C
B	C	A
C	A	B

A	B	C	D
B	A	D	C
B	D	A	C
B	C	D	A
C	A	D	B
C	D	A	B
C	D	B	A
D	A	B	C
D	C	A	B
D	C	B	A

6.2. ábra. A lehetséges cserék $s = 2, 3$ és 4 elem esetén, ha egyik elem sem marad helyben.

A bizonyítás megtalálható például Király és Tóth (2011) jegyzetében.

A D_n sorozat következő, egymást követő értékei $D_5 = 44, D_6 = 265, D_7 = 1854$, stb. Magukat a lehetséges cseréket továbbra is viszonylag könnyen megkonstruálhatjuk az előző lépésekben felírt cserék felhasználásával[4], ugyanakkor látható, hogy a lehetséges cserék száma meglehetősen gyorsan nő. Mivel $\lim_{s\to\infty} \frac{D_s}{s!} = \frac{1}{e}$, ezért s elem esetén a megengedett cserék száma $\Theta(s!)$ nagyságrendű. Ha még azt is figyelembe vesszük, hogy

[4]A lehetséges cserék felírását adja a következő konstrukció. Tegyük fel, hogy az $A\ B\ C\ D\ E\ldots$, n darab elemet kell felcserélnünk úgy, hogy egyik sem maradhat a helyén. Az általánosság megszorítása nélkül tegyük fel, hogy az első helyre az új besorolásban a B kerül. Ekkor két lehetőségünk van:
1, Az A a B helyére kerül, amely esetben a többi elemet D_{n-2}-féleképpen cserélhetjük fel úgy, hogy senki sem kerül ugyanoda.
2, Az A nem a B helyére kerül, hanem a $C, D, E, \ldots$ helyek valamelyikére. A példa kedvéért tegyük fel, hogy ez a C hely. Ekkor ugyancsak két esetet kell meggondolnunk: i) a C-t beírjuk a B helyére, így a többi elemet D_{n-3}-féleképpen rendezhetjük, vagy ii) úgy tekintjük, hogy a C nem kerülhet a B helyére, ami pedig D_{n-2} lehetséges cseréhez vezet.
Összegezve a lehetőségeket $(n-1)\left[(n-1)D_{n-2} + D_{n-3}\right]$ esetet kapunk, amely $n \geq 3$-ra megegyezik D_n-nel.

a rendelkezésre álló k darab m-fős csoportból hányféleképpen választhatunk ki s cserére jelölt elemet, akkor láthatjuk, hogy ha az LCW algoritmus metódusát alkalmaznánk, akkor minden egyes javító lépésnél - feltéve, hogy egy csoportot és egy elemet rögzítettnek tekintünk - összesen

$$\binom{k-1}{s-1} m^{s-1} D_s$$

esetet kellene figyelembe vennünk, ami már kis k és m értékeknél is vállalhatatlanná válik.

Így a fentiek miatt az algoritmus továbbgondolásánál a kettes cserék mellett csupán a hármas cseréket vesszük figyelembe. Ezt háromféleképpen tesszük meg, így a hármas cserékre három különböző heurisztikát konstruálunk. Az első változat (`LCWv2`) során, amelynek pszeudokódját a 6.4. Algoritmus írja le, miután megpróbáltunk kettes cserékkel javítani a szobák össztávolságának értékén, ugyenezt megtesszük hármas cserékre szorítkozva is.

LCW Algoritmus második verzió - hármas cserék (`LCWv2`)

(1 - 3) ugyanaz, mint az LCW algoritmus esetében

(4) *Ha történt csere a legutóbbi iteráció során, akkor újra végigmegyünk az összes elemen.*

if (Flag == true)

Flag $\leftarrow$ false, $i \leftarrow 0$

goto Step (3)

end if

(5) *Megnézzük, hogy lehet-e javítani hármas cserével úgy, hogy az i elemet és két másik, különböző csoportokban lévő elemet felcserélünk.*

$i \leftarrow i + 1$

if ($i \leq N$)

$t \leftarrow$ az a csoport, amelyben az i elem van, vagyis amelyre $x_{it} = 1$

$(k_1, k_2) \leftarrow \underset{(j_1,j_2)\in J\cdot J}{\arg\max}\ W(j_1, j_2)$, ahol

$W(j_1, j_2) = r_{iq_2} + r_{j_1 r} + r_{j_2 q_1} - (r_{it} + r_{j_1 q_1} + r_{j_2 q_2} + d_{ij_1} + d_{ij_2} + d_{j_1 j_2})$

és

$J \cdot J = \{(j_1, j_2) \mid x_{j_1 q_1} = 1, 1 \leq q_1 \leq G, q_1 \neq t,$

$x_{j_2 q_2} = 1, 1 \leq q_2 \leq G, q_2 \neq t, q_1 \neq q_2\}$

if $(W(k_1, k_2) > 0)$

[kicseréljük a három elemet]

$x_{k_1 q_1} \leftarrow 0, x_{k_1 t} \leftarrow 1, x_{k_2 q_2} \leftarrow 0, x_{k_2 q_1} \leftarrow 1, x_{iq_2} \leftarrow 1, x_{it} \leftarrow 0$

$R \leftarrow DX$ [frissítjük az R mátrixot]

Flag $\leftarrow$ true [jelöljük, hogy csere történt]

end if

goto Step (5)

end if

(6) *Megnézzük, hogy történt-e csere a legutóbbi iteráció során, és ha nem, akkor az algoritmus terminál.*

if (Flag == false)

stop [nem lehet tovább javítani cserékkel]

else

Flag $\leftarrow$ false, $i \leftarrow 0$

goto Step (5)

end if

6.4. Algoritmus. LCW Algoritmus második verzió - hármas cserék (`LCWv2`)

6.2.3. Megjegyzés. Vegyük észre, hogy a 6.4. Algoritmus során a hármas cserékkel történő javításnál egy négyzetes mátrix (a főátlóban 0-k állnak) maximális elemét keressük, amelynek köszönhetően a javítási lépés könnyedén implementálható.

Az LCW algoritmus hármas cserékkel kiegészített második változatában (`LCWv3`) első lépésben kettes cserékkel próbálunk meg javítani, majd ha ezzel már nem tudunk, akkor a hármas cserék következnek. Ezt a két fázist egészen addig végezzük el egymás után ismételten, amíg már sem kettes, sem hármas cserékkel nem tudtunk javítani.

Végül a harmadik változatban (LCWv4) minden egyes javítási lépésnél egyszerre tekintjük a lehetséges kettes, illetve hármas cseréket. Ezek közül azt a lehetőséget választjuk, amellyel a legtöbbet javítunk. Ha kettes és hármas cserével is ugyanakkora mértékű javulást érnénk el, akkor a kettes cserét preferáljuk. Az algoritmus akkor terminál, ha semelyik elem esetén sem lehet már kettes vagy hármas cserével javítani.

6.2.4. LCW tabu kereséssel és stratégiai ingázással (TLCW, SO)

Gallego és szerzőtársai (2013) a *maximálisan diverz csoportosítási problémát* (maximally diverse grouping problem, MDGP) vizsgálják. Tanulmányukban tabu keresést, valamint stratégiai oszcillációt alkalmazó módszereket konstruálnak meg és tesztelnek.

Az MDGP feladat lényegében a Weitz és Lakshminarayanan (1996, 1998) által tárgyalt probléma (lásd a 6.2.2. Fejezetet) kiterjesztésének tekinthető, ahol egy csoport diverzitását a csoporton belüli elemek páronkénti távolságainak összege adja. Gallego és szerzőtársai (2013) az MDGP probléma két variánsát tárgyalja. Az elsőben minden csoport elemszámának azonosan $S = N/G$-nek kell lennie, amely a Weitz és Lakshminarayanan (1996, 1998) tanulmányában szereplő problémával ekvivalens. Ezt a problémát MDGP1-ként jelölik. A második ennek általánosítása, amelyben bármely $g \in \{1, \ldots, G\}$ csoport S_g elemszáma egy $[a_g, b_g], a_g \leq b_g$ intervallumba kell, hogy essen. Ezt MDGP2-ként nevezik el, elemzésük során pedig a vizsgált algoritmusokat mindkét változaton tesztelik.

Gallego és szerzőtársai (2013) egy úgynevezett *multi-start* (többször újrakezdő) elemzési keretet használ, amelynek lényege, hogy az algoritmusok különböző állapotokból újraindulva a lehetséges megoldások újabb és újabb tartományait térképezik fel. A szerzők az általuk vizsgált algoritmusok konstrukciójához három alapvető építőelemet használnak fel: (1) a kezdeti megoldás megkonstruálása, (2) javító módszer szomszédság kereséssel és (3) stratégiai ingázás. Ezek különböző kombinációját tesztelik, ugyanakkor nem minden esetben jelenik meg az adott algoritmusban mindhárom elem.

Kezdeti megoldás megkonstruálása

A kezdeti megoldás meghatározsához használt *mohó konstrukció* (greedy construction, GC) pszeudokódját a 6.5. Algoritmus írja le. Ennek első lépéseként G elem véletlenszerű kiválasztásával inicializálnak G darab egyelemű csoportot, majd ezt második és harmadik lépésekként a következő két fázis követi. Az első fázisban a cél azon csoportok feltöltése

elemekkel, amelyeknek elemszáma még nem éri el a megadott alsó elemszámkorlátot. A második fázisban pedig a felső elemszámkorlát figyelembevételével csoportosítják a megmaradt elemeket. Mindkét fázis során iteratív lépések révén véletlenszerűen választanak egy i elemet, majd azt ahhoz a g csoporthoz rendelik hozzá, amelyre a csoport elemeitől vett átlagos távolság, vagyis a

$$D_{ig} = \frac{\sum_{j \in E_g} d_{ij}}{|E_g|}$$

érték maximális, ahol E_g jelöli a g csoporthoz aktuálisan hozzárendelt elemek halmazát, d_{ij} pedig az i és j elemek távolságát. A második és harmadik lépés során $N - G$ iteráció történik.

Az egyszerű GC konstrukció mellett Gallego és szerzőtársai (2013) ennek két további változatát is bevonja az elemzésbe. Az egyikben a GC algoritmust a Mingers és O'Brien (1995) által is tárgyalt FULL módszerrel ötvözik (GC-FULL), míg a másikban a tabu keresésből átvett elemeket alkalmaznak (GC-Tabu). A GC-FULL a 6.5. Algoritmus (2)-es és (3)-as lépésein belül a hozzárendeléshez megkeresi azt az (i, g) elem-csoport párt, amely maximalizálja a D_{ig} értéket. A GC-Tabu arra épít, hogy a kezdeti megoldások konstruálása egy multi-start keretben történik. Ez két struktúra segítségével rögzíti a korábbi megoldásokhoz kapcsolódó releváns információkat, majd ezt felhasználja az új megoldások meghatározásánál. Ezek az információk bármely két elemre a következőek: hányszor voltak ugyanabba a csoportba beosztva a korábbi megoldások során, valamint ezen megoldások átlagos minősége, vagyis célfüggvényértéke. Ezen statisztikák figyelembevételével a 6.5. Algoritmus GC-Tabu variánsában a D_{ig} távolságérték úgy módosul, hogy az elősegítse olyan elempárok csoportjainak a kialakulását, amelyek alacsony gyakorisággal és magas minőséggel jártak együtt a korábbiakban.

Mohó konstrukció (Greedy Construction, GC, Gallego és szerzőtársai (2013))

---

(1) Véletlenszerűen választunk G elemet és mindegyiket egy-egy csoporthoz rendeljük

(2) A következő lépéseket ismételjük, amíg $|E_g| \geq a_g$ nem teljesül bármely g csoportra

   (a) Véletlenszerűen választunk egy nem csoportosított i elemet

(b) Az i elemet ahhoz a g csoporthoz rendeljük, amelyre $|E_g| < a_g$ és D_{ig} maximális

(3) A következő lépéseket ismételjük, amíg van nem csoportosított elem

(a) Véletlenszerűen választunk egy nem csoportosított i elemet

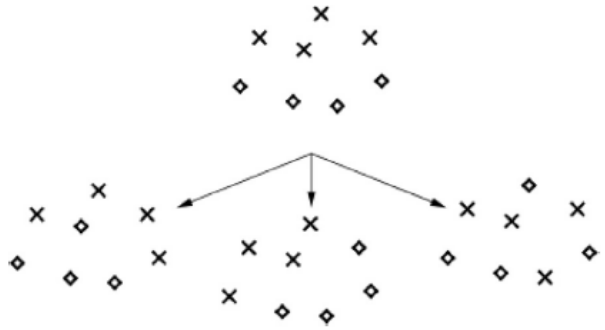
(b) Az i elemet ahhoz a g csoporthoz rendeljük, amelyre $|E_g| < b_g$ és D_{ig} maximális

---

6.5. Algoritmus. Mohó konstrukció (Greedy Construction, GC, Gallego és szerzőtársai (2013))

Javító módszerek áthelyezésekkel és tabu memóriával

A kezdeti megoldások javítására Gallego és szerzőtársai (2013) minden esetben a csere szomszédság keresésén alapuló módszert alkalmaznak. Baker és Powell (2002) nyomán egy adott csoportosítás *csere szomszédsága* (swap neighbourhood[5]) azon lehetséges csoportosítások halmaza, amelyeket úgy kapunk, hogy a kiinduló állapothoz képest két különböző csoportban lévő i és j elemet felcserélünk. A 6.3. ábra mutatja egy egyenletes MSSC megoldás három lehetséges szomszédját, ha két klaszter van a csoportosításban.



6.3. ábra. Egy egyenletes MSSC megoldás három lehetséges szomszédja a csere szomszédságban. Forrás: Costa és szerzőtársai (2017)

A Gallego és szerzőtársai (2013) által tárgyalt javító algoritmusok lépésenként javítanak az aktuális megoldáson úgy, hogy egy előre meghatározott szempont szerint egy olyan csoportosítást választanak a csere szomszédságból, amely javít a célfüggvény értékén. A szerzők többféle előre meghatározott szempontot is említenek. Fan és szerzőtársai (2011)

[5]Gallego és szerzőtársai (2013) egyszerűen a 'szomszédság' kifejezést vezeti be, ugyanakkor az egyértelműség kedvéért mi a dolgozatban általánosan a Costa és szerzőtársai (2017) által használt 'csere szomszédság' elnevezést alkalmazzuk, amelyet esetenként egyszerűsítünk.

a *legjobb javítás* (best improvement, BI) stratégiát alkalmazza, amely során a szomszédság összes lehetséges csoportosítása közül azt választják ki, amely a legnagyobb mértékben javítja a célfüggvényértéket. Egy másik lehetséges módszer az *első javítás* (first improvement, FI), melynek lényege, hogy a szomszédság feltérképezése során az első olyan csoportosítást választjuk, amely növeli a célfüggvény értékét. További lehetőséget adnak meg az LC és az LCW módszerek (lásd a 6.2.2. alfejezetet) is, amelyek végigiterálnak az összes elemen, és az iteráció minden egyes lépésében egy adott elem valamilyen lehetséges cseréit tekintik. Akár javítunk, akár nem, az iteráció a következő elemmel folytatódik, és mindaddig újraindul az első elemtől, amíg az összes elemet végignézve történt csere.

A szakirodalomban általában olyan tanulmányokkal találkozunk, ahol a csoportok elemszámára nem alsó és felső korlátot írnak elő, hanem előre meghatározott értékeket, legyen szó akár egyenletes klaszterezésről, vagy sem. Ekkor a célfüggvény értékén párok cseréjével (*swap* vagy *switch*) próbálnak meg javítani, hogy a megoldások a keresés alatt végig megengedettek maradjanak. Gallego és szerzőtársai (2013) problémafelvetése ugyanakkor lehetővé teszi azt is, hogy a keresés során *áthelyezéseket* (insertion) is figyelembe vegyenek, amelyek során csupán egy elem csoport-hozzárendelését változtatják meg.

Emellett tanulmányukban egy rövid távú tabu memóriát is bevezetnek, amely révén a keresés még azután is folytatódik, hogy az elért egy lokális optimumot. Ekképpen, amikor a javító módszer már nem tud javító lépést végrehajtani, a javító módszernek megfelelő módon a legjobb nem-javító lépést választják. Az ekkor áthelyezett elemeket ezután $tabuTenure$ iteráción keresztül nem mozgatják. Az eljárás pedig akkor terminál, amikor $maxTenure$ iteráció óta nem történt javítás a korábban talált legjobb megoldáshoz képest.

Gallego és szerzőtársai (2013) a beillesztéseket és a tabu memóriát három javító módszer, a BI, az FI és az LCW esetében alkalmazzák, az így kapott algoritmusokat rendre T-BI, T-FI és T-LCW jelöli. A legjobb kezdeti megoldás és javító módszer kombinációjának kiválasztására a következő algoritmus párokat tesztelik. A kezdeti megoldás konstrukciójára mindhárom mohó variánst (GC, GC-FULL, GC-Tabu) figyelembe veszik, valamint negyedik lehetőségként a WJ (Weitz-Jelassi, lásd a 6.2.2. alfejezetet) algoritmust alkalmazzák. A javító módszerek közül a tabu memóriával ellátottakat és azok tabu memória nélküli változatait veszik be az elemzésbe. Előzetesen elvégzett kísérletek

alapján a *tabuTenure* értékét 0,1M-ra, míg a *maxTenure* értékét 0,5M-ra állították be. Az elvégzett kísérletek során a 'kezdeti megoldás'-'javító módszer' párok közül a legjobb eredményt az egyszerű GC konstrukció adta a T-LCW javító módszerrel.

6.2.4. Megjegyzés. Gallego és szerzőtársai (2013) nem tesz róla említést, de a keresés során előfordulhat, hogy a tabu memória révén már a lehetséges további *maxTenure* iteráció előtt elfogynak az áthelyezhető vagy felcserélhető elemek. Az algoritmus ekkor is terminál.

Stratégiai ingázás

A *stratégiai ingázás* (strategic oscillation, OS, lásd még Glover és Laguna (1997)) lényege, hogy az algoritmus olyan keresési tartományt is feltérképezzen, amely a probléma felírása szerint nem-megengedett megoldásokat tartalmaz. A megkonstruált algoritmus esetében az ingázást egy egészértékű k paraméteren keresztül vezetik be, amelynek értéke 0 és egy előre meghatározott $k_{\max}$ érték között változhat. Ezt pedig a csoportok elemszámára vonatkozó korlátok módosítására, pontosabban az intervallum bővítésére alkalmazzák, ekképpen

$$a_g - k \leq |E_g| \leq b_g + k.$$

Ez tehát azt jelenti, hogy $k > 0$ esetén az algoritmus olyan megoldásokat is találhat, amelyek a probléma eredeti felírása szerint nem megengedettek. Az ingázást az valósítja meg, hogy amikor a stratégiai ingázás alkalmazása közben egy javító megoldást találnak, akkor a k értékét visszaállítják 1-re. A stratégiai ingázás pszeudokódját a 6.6. Algoritmus írja le. Ez a kezdeti megoldás bármelyik konstrukciójával, valamint bármely javító módszerrel kombinálható. Az (5)-ös lépésben a nem megengedett megoldás helyreállítása Fan és szerzőtársai (2011) LSGA algoritmusában is alkalmazott mechanizmussal történik. Ennek megfelelően eltávolítunk véletlenszerűen kiválasztott elemeket olyan csoportokból, amelyekre $|E_g| > b_g$ (vagy amennyiben ilyen csoportok nincsenek, akkor $|E_g| > a_g$), és áthelyezzük őket olyan csoportokba, amelyekre $|E_g| < a_g$. Ezt addig ismételjük, amíg a csoportok elemszámai újra megengedettek nem lesznek. Az elvégzett elemzés során Gallego és szerzőtársai (2013) a $k_{\max} = 4$ értéket találta a legjobbnak.

Stratégiai ingázás (Strategic Oscillation, SO, Gallego és szerzőtársai (2013))

while (tart a keresési idő)

(1) Megkonstruálunk egy kezdeti megengedett megoldást

(2) $k \leftarrow 0$

$s \leftarrow$ valamely javító módszer alkalmazásával kapott megoldás

(3) $k \leftarrow 1$

while $(k \leq k_{\max})$

(4) $s' \leftarrow$ az s megoldásra alkalmazzunk egy javító módszert, miközben az $a_g - k \leq |E_g| \leq b_g - k$ elemszámkorlátot írjuk elő a csoportokra

(5) Állítsuk helyre az s' megoldást, ha nem megengedett

Alkalmazzuk a javító módszert $k = 0$ mellett (vagyis $a_g \leq |E_g| \leq b_g$)

(6) **if** (s' jobb, mint s)

$s \leftarrow s', k \leftarrow 1$

else

$k \leftarrow k + 1$

end if

end

end

6.6. Algoritmus. Stratégiai ingázás (Strategic Oscillation, SO, Gallego és szerzőtársai (2013))

Fan és szerzőtársai (2011) LSGA módszere az MDGP általános verziójára alkalmazható, és egy ún. *hibrid genetikai* (hybrid genetic) algoritmus, amely egy genetikai algoritmust és egy lokális keresés eljárást ötvöz. Az LSGA véletlenszerűen generált megengedett megoldások egy kezdő populációjából indul ki, amelyet a következő módon frissít. A módszer a megengedett megoldások célfüggvényértékével arányos valószínűséggel kiválaszt egy pár 'szülő' megengedett megoldást, ezeket egy új 'utód' megoldássá kombinálja, amelyet helyreállít, ha az nem megengedett, majd megpróbál javítani is rajta, így egy új megengedett megoldást konstruálva. Ezt a lépést előre meghatározott számszor elvégezzük, majd az így kapott, valamint a kezdő populációban lévő megengedett megoldások közül kiválaszt-

juk az előre meghatározott számú legjobbat, ezzel frissítve a populációt. Az iteratív lépést addig ismételjük, amíg egy megállási feltétel nem teljseül.

Gallego és szerzőtársai (2013) az elemzésük során 480 darab, véletlenszerűen generált mintát tekint, összesen 48 eltérő kategóriával, mindegyikben 10-10 mintával.[6] A kategóriákra különböző időkorlátok (1-600 másodperc), elemszámok (10-960), csoportszámok (2-24), csoportokra vonatkozó elemszámkorlátok (rögzített, valamint alsó és felső korlátok) és elemek közötti távolságok vonatkoznak. Az elemek közötti távolságokra három különböző verziót tekintenek, melyek mindegyikében ugyanannyi minta szerepel. Az elsőben (RanReal) a távolságokat az $U(0, 100)$ egyenletes eloszlásból generálják. A másodikban (RanInt) a távolságok az $U(0, 100)$ diszkrét egyenletes eloszlásból származnak. A harmadik (Geo) esetén az elemek koordinátáit generálják véletlenszerűen, és az elemek közötti távolságokat az euklideszi távolságok adják. Ehhez a dimenziók számát véletlenszerűen választják 2 és 21 között, az elemek koordinátáit pedig a $[0, 10]$ intervallumból generálják.

Az elvégzett elemzés utolsó lépésében Gallego és szerzőtársai (2013) az LSGA algoritmust, az LCW heurisztikát (véletlenszerű újrakezdésekkel), a T-LCW eljárást (GC konstrukcióval) és az SO módszert (GC konstrukcióval és T-LCW javításokkal) hasonlította össze. Az egyes algoritmusokat addig futtatták újra és újra, amíg a kategóriára előírt időkorlátot el nem érte azok futásideje. A kísérletekben a RanReal és RanInt minták esetén az SO, míg a Geo minta esetén a T-LCW módszer teljesített a legjobban. Emiatt a 7. Fejezetben bemutatott elemzésbe mi mind a T-LCW, mind az SO módszert bevesszük, és rendre `TLCW`, valamint `SO` névvel hivatkozunk rájuk.

6.2.5. Malinen és Fränti algoritmusa (`MalinenFranti`)

Malinen és Fränti (2014) olyan *egyenletes k-közép* (balanced k-means) eljárást alkalmaznak, amelyben a k-közép eljárásnak megfelelő hozzárendelési lépést párosítási problémaként írják fel, amelyet a Magyar módszer (Kuhn, 1955) segítségével $\mathcal{O}(n^3)$ időben oldanak meg, ahol n jelöli az elemek számát.

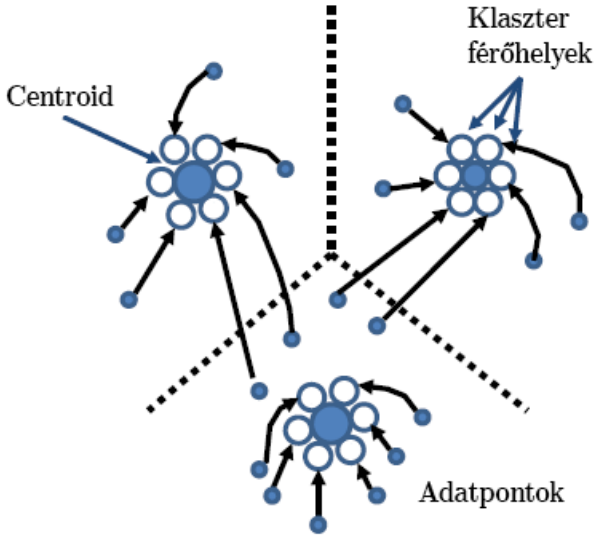
A hozzárendelési feladat formális definícióját a következőképpen adják meg. Legyenek adva A és S, egyenlő méretű halmazok, és egy $W : A \times S \to \mathbb{R}$ súlyfüggvény. A cél, hogy

[6] A Gallego és szerzőtársai (2013) által generált minta letölthető a `https://grafo.etsii.urjc.es/optsicom/mdgp/` oldalról.

egy olyan $f : A \to S$ bijekciót találjunk, amely révén a

$$\text{Cost} = \sum_{a \in A} W(a, f(a))$$

költségfüggvény értéke minimális. A javasolt algoritmus kontextusában az A és S halmazok klaszterférőhelyeknek és adatpontoknak felelnek meg (lásd a 6.4. ábrát).



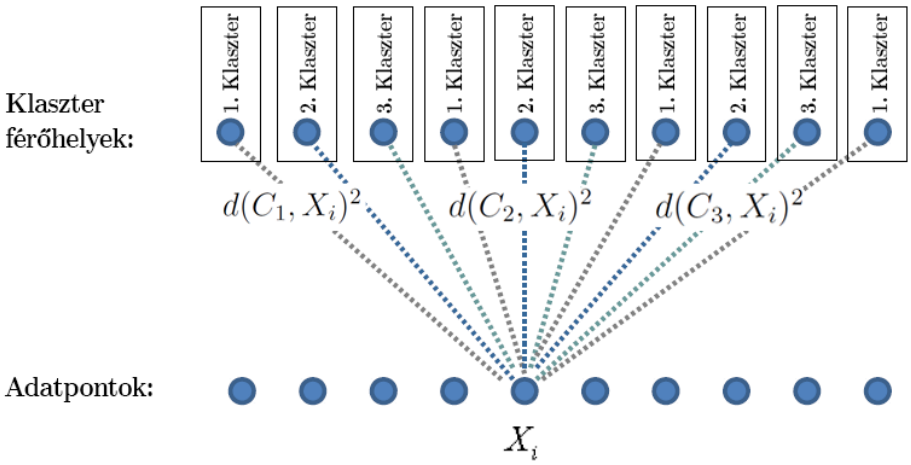
6.4. ábra. Pontok hozzárendelése centroidokhoz klaszterférőhelyek által. Forrás: Malinen és Fränti (2014).

Malinen és Fränti (2014) egyenletes k-közép algoritmusa majdnem megegyezik a k-közép algoritmussal, az egyetlen eltérés a hozzárendelési fázisban jelenik meg. Ahelyett, hogy a legközelebbi centroidokat választanák, n előre megadott férőhelyet határoznak meg (klaszterenként n/k-t), és az adatpontokat csak ezekhez lehet hozzárendelni (lásd a 6.4. ábrát). Ekképpen minden klaszter egyenlő méretű lesz, feltéve, hogy $\lceil n/k \rceil = \lfloor n/k \rfloor = n/k$. Máskülönben ($n \mod k$) darab $\lceil n/k \rceil$ méretű, és $k - (n \mod k)$ darab $\lfloor n/k \rfloor$ méretű klaszter keletkezik.

A cél annak a hozzárendelésnek a megtalálása, amely minimalizálja az *átlagos négyzetes hibát* (mean squared error, MSE), vagyis az

$$\text{MSE} = \sum_{j=1}^{k} \sum_{X_i \in C_j} \frac{\|X_i - C_j\|^2}{n},$$

értékét, ahol X_i egy adatpontot jelöl, C_j pedig egy centroidot. Ehhez először konstruálnak egy páros gráfot, amely n adatpontot és n klaszter férőhelyet tartalmaz (lásd a 6.5. ábrát). Második lépésként annyira egyenletesen osztják el a klaszter férőhelyeket a klaszterek között, amennyire csak lehetséges.



6.5. ábra. Minimális MSE számítása egyenletes klaszterek esetén, páros gráffal modellezve. Forrás: Malinen és Fränti (2014).

Ezután centroidokat rendelnek a particionált férőhelyekhez, klaszterenként pontosan egyet. A kezdeti centroidok véletlenszerűen is választhatóak az összes adatpont közül. Az éleken lévő költségek megegyeznek a pontok és a klasztercentroidok között lévő távolságok négyzetével. A hagyományos hozzárendelési problémával ellentétben - ahol az élsúlyok állandóak - itt a súlyok dinamikusan változnak minden egyes k-közép iteráció után az újonnan számított centroidok alapján.

Következő lépésként a Magyar módszer segítségével meghatározzák a minimális súlyú párosítást. Ehhez a négyzetes távolságokat egy $n \times n$-es mátrixban tárolják. A frissítési lépés hasonlít a k-közép algoritmuséhoz, ahol a centroidokat a klaszterhez rendelt adatpontok átlagaként számítják:

$$C_i^{(t+1)} = \frac{1}{n_i} \sum_{X_j \in C_i^{(t)}} X_j. \tag{6.8}$$

Az élsúlyok frissítése közvetlenül a centroidok frissítése után történik.

Az így kapott eljárás pszeudokódját a 6.7. Algoritmus írja le. Az élsúlyok számításánál a klaszter férőhely sorszámát a jelöli, és a $\bmod$ függvény segítségével határozzuk meg, hogy egy adott klaszter férőhely melyik klaszterhez tartozik. Így az X_i adatpont és az i-edik klaszter férőhely távolsága:

$$W(a,i) = \operatorname{dist}\left(X_i, C^t_{(a \mod k)+1}\right)^2 \quad \forall a \in [1,n] \quad \forall i \in [1,n]. \tag{6.9}$$

Miután konvergált az algoritmus az $X_i \in [1,n]$ pontok $P = \{P_1, \ldots, P_k\}$ particiója az

alábbi szerint adódik:

$$X_{f(a)} \in P_{(a \mod k)+1}.$$

Az algoritmusban a domináns lépés a minimális súlyú párosítás megtalálása, amely a Magyar módszer segítségével $\mathcal{O}(n^3)$ időben oldható meg.

Malinen és Fränti (2014) az algoritmusukat Bradley és szerzőtársai (2000) *korlátozott k-közép* (constrained k-means) eljárásával hasonlítják össze. Az elemzést mesterséges adathalmazokon, valamint a való életből származó, 'Thyroid', 'Wine' és 'Iris' nevű adatokon végzik el.[7] Az adathalmazok méretei 150-5000-ig terjednek. Általnosságban a korlátozott k-közép módszer kicsivel jobb eredményeket adott a célfüggvényérték tekintetében, ugyanakkor a futásidő szempontjából az egyenletes k-közép eljárás múlta felül versenytársát. Ezt mutatja az is, hogy az 5000 elemet tartalmazó esetben csupán az egyenletes k-közép módszert tudták lefuttatni (ezt egyszer tették meg 1 óra 40 perces futásidővel), a korlátozott k-közép módszer pedig 1 nap alatt sem terminált.

Az algoritmus MATLAB kódját a szerzők nyilvánosan elérhetővé tették, így az letölthető a `https://archive.uef.fi/en/web/machine-learning/software/` címen (Balanced.zip). A módszer teszteléséhez ezt alkalmaztuk, az elemzésben pedig `MalinenFranti` néven hivatkozunk rá.

Malinen-Fränti (MF) Algoritmus (Malinen és Fränti, 2014)

---

(1) *Inicializálás*

$C^0 \leftarrow$ kezdő klaszter centroidok inicializálása.

$t \leftarrow 0$

(2) *Hozzárendelési lépés*

$W(a,i)\ \forall a \in [1,n]\ \forall i \in [1,n] \leftarrow$ élsúlyok számítása (6.9) alapján

$X_{f(a)}\ \forall a \in [1,n] \leftarrow$ hozzárendelési probléma megoldása a Magyar módszerrel

(3) *Frissítési lépés*

$C^{t+1} \leftarrow$ új centroid lokációk számítása (6.8) alapján

if $(C^{t+1} \neq C^t)$ [ha a centroidok változtak]

[7] Az adatok megtalálhatóak a `http://cs.uef.fi/sipu/datasets/` címen.

$t \leftarrow t + 1$

goto Step (2)

end if

6.7. Algoritmus. Malinen-Fränti (MF) Algoritmus (Malinen és Fränti, 2014)

6.2.6. "A Kevesebb Több" Megközelítés - Változó Szomszédság Keresés (LIMA-VNS) Algoritmus (`Costaetal`)

Costa és szerzőtársai (2017) egy sztenderd *Változó Szomszédság Keresés* (Variable Neighborhood Search, VNS, lásd Hansen és Mladenović (2001b)) algoritmust mutat be az egyenletes MSSC klaszterezésre, amely az újszerű *"A Kevesebb Több" Megközelítésre* ("Less Is More" Approach, LIMA, lásd Mladenović és szerzőtársai (2016)) épít.

A szerzők felírása szerint $P = \{p_1, p_2, \ldots, p_n\}$ n darab adatpont adott egy $\mathbb{R}^S$ euklideszi térben. A cél k klaszter meghatározása, amely révén az adatpontok és az egyes adatpontokhoz tartozó klaszterközéppont négyzetes távolságának összege minimális. Így az optimalizálási feladat az alábbiak szerint írható le:

$$\begin{aligned} &\min_{w,y} \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} \|p_i - y_j\|^2 \\ &\text{s.t.} \sum_{j=1}^{k} w_{ij} = 1, \ \forall i = 1, \ldots, n \\ &\qquad w_{ij} \in \{0, 1\}, \ \forall i = 1, \ldots, n, \forall j = 1, \ldots, k, \end{aligned}$$

ahol az $y_j \in \mathbb{R}^S, j = 1, \ldots, k$ döntési változók adják a k klaszterközéppontot, míg a w_{ij} döntési változók fejezik ki, hogy a p_i pontot a j-edik klaszterhez rendeltük-e hozzá. Az azonos csoportelemszámok elérésére a szerzők a fenti optimalizálási problémához a következő korlátot adják hozzá:

$$\sum_{i=1}^{n} w_{ij} = \frac{n}{k}, \forall j = 1, \ldots, k,$$

amikor n a k többszöröse, egyébként pedig lesz ($n \mod k$) klaszter, amelynek mérete $\lceil \frac{n}{k} \rceil$, és lesz $k - (n \mod k)$ klaszter, amelynek mérete $\lfloor \frac{n}{k} \rfloor$. A probléma megoldására

alkalmazott VNS metaheurisztikus eljárást és a LIMA módszert Costa és szerzőtársai (2017) cikke alapján mutatjuk be.

A VNS alapjai

A Változó Szomszédság Keresés (VNS) egy sztochasztikus kereső módszer, amelynek célja az optimális vagy közel optimális megoldás megtalálása globális optimalizálási problémákban. Ez egy metaheurisztikus eljárás, vagyis egy olyan keret, amely segít más heurisztikáknak kijutni lokális optimumokból. A VNS előnyeit támasztja alá, hogy sikeresen használták számos NP-nehéz klaszterezési probléma során (Aloise és szerzőtársai, 2013; Belacel és szerzőtársai, 2002; Hansen és Mladenović, 2001a; Hansen és szerzőtársai, 2012; Santi és szerzőtársai, 2016).

Az egyenletes MSSC a kombinatorikai optimalizálási problémák osztályába tartozik, amely az általánosság megszorítása nélkül minimalizálási problémaként a következőképpen adható meg. Legyen $Z = \{1, \ldots, z\}$ indexek egy véges halmaza, és legyen $c = (c_1, \ldots, c_z)$ egy z-vektor. Legyen $F \subseteq Z$ esetén $c(F) = \sum_{j \in F} c_j$. Tegyük fel, hogy adott a Z részhalmazainak egy $\mathcal{F}$ halmaza. A *kombinatorikai optimalizálási probléma* ekkor

$$\min\{c(F) : F \in \mathcal{F}\}.$$

A klaszterezési probléma esetében a c_j értékek jelölik azon klaszterek költségeit, amelyek valamely csoportosításban előfordulnak. A klaszterezési probléma egy megengedett megoldása olyan csoportokból áll, amelyek az összes elemet lefedik, ezen csoportok j indexek halmaza egy F halmaz. Végül pedig az F által meghatározott klaszterezés költségét $c(F)$, vagyis a c_j értékek összege adja.

Egy adott $C = (Z, \mathcal{F}, c)$ kombinatorikai optimalizálási probléma esetén definiáljuk minden egyes $F \in \mathcal{F}$ megengedett megoldás karakterisztikus vektorát úgy, mint $\chi^F \in \{0,1\}^Z$, ahol $\chi^F_j = 1$, ha $j \in F$, és $\chi^F_j = 0$ egyébként. Így C-re úgy is tekinthetünk, mint egy konvex politópon vett minimalizálási probléma[8], vagyis

$$\min\left\{c^T x | x \in \operatorname{conv}\left\{\chi^F \in \{0,1\}^Z | F \in \mathcal{F}\right\}\right\}. \tag{6.10}$$

[8]A konvex politóp az $\left\{\chi^F \in \{0,1\}^Z | F \in \mathcal{F}\right\}$ halmaz konvex burkaként áll elő, amely tartalmazza az összes olyan pontot, amely felírható a halmazbeli pontok konvex kombinációjaként. A minimalizálási probléma megoldása egy csúcson vétetik fel, így az egy megengedett megoldás.

Ekkor a (6.10) egy x^* lokális minimuma egy olyan vektor, amelyre

$$c^T x^* \leq c^T x, \forall x \in \mathcal{N}(x^*),$$

ahol $\mathcal{N}(x)$ jelöli az x szomszédainak halmazát, vagy másképpen *szomszédságát* (neighborhood). A kombinatorikai optimalizálási problémákban egy x megoldás szomszédait számos különböző módon megkaphatjuk, például az x egy vagy több komponensének komplementerét véve. A VNS egy optimalizálási problémára számos szomszédságot szisztematikusan végigpásztáz, hogy megtalálja a globális optimumot. Az alapötlet a VNS mögött, hogy egy adott szomszédság lokális minimuma nem feltétlenül az egy másik szomszédságban. Következésképpen, a szomszédságok változtatása hatékony lehet a lokális minimumtól történő elszakadásban. Persze egy globális minimum lokális minimum függetlenül attól, hogy épp melyik szomszédságot tekintjük.

A VNS egy *egyetlen trajektóriájú* (single trajectory) metaheurisztika, mint a *szimulált hűtés* (simulated annealing) és a tabu keresés (a metaheurisztikus optimalizálási eljárásokról például Boussaïd és szerzőtársai (2013) ad áttekintést). Ez azt jelenti, hogy egy x megoldást az algoritmus teljes futásideje alatt nyilvántartunk. A fő különbség a többi metaheurisztikus kerethez képest, hogy a javításokat az x folyamatosan növekvő szomszédságaiban keressük. Először az aktuális $\mathcal{N}_t(x)$ szomszédságból véletlenszerűen kiválasztunk egy x' szomszédos megoldást, és ezután egy *lokális leereszkedési módszert* (local descent method) alkalmazunk x'-ből kiindulva, amely végén egy újabb x'' lokális minimumhoz jutunk. Ha x'' rosszabb (vagy ugyanannyira jó), mint x, akkor figyelmen kívül hagyjuk, és az algoritmus egy új, véletlenszerű x' megoldásból indul ki, amelyet az x egy szélesebb szomszédságából választunk ki, vagyis $\mathcal{N}_{t+t_{step}}(x)$-ből. Ellenkező esetben pedig x'' helyettesíti x-et, és az algoritmus az új legjobb megoldás legszűkebbb szomszédságából kiindulva folytatódik. Valahányszor elérjük a legtágabb szomszédságot, vagyis $\mathcal{N}_{t_{max}}$-ot, a VNS újraindul a legszűkebb szomszédságból egészen addig, amíg valamilyen megállási kritérium nem teljesül (például el nem érjük a megadott maximális CPU időt, vagy egy rögzített ismétlésszámot).

LIMA-VNS heurisztika az egyenletes MSSC problémára

Általánosságban véve, a VNS különböző szomszédsági struktúrákat használhat mind a *szélesítési* (diversification), mind a *mélyítési* (intensification) lépésekben (lásd például Aloise

és szerzőtársai (2006), valamint Ribeiro és szerzőtársai (2008)). Ugyanakkor nemrég született tanulmányok rámutattak, hogy egy bizonyos szomszédsági struktúrán alapuló VNS heurisztika különösen hatékony lehet (lásd Brimberg és szerzőtársai (2017), Mladenović és szerzőtársai (2016)), amely az úgynevezett *"a kevesebb több" megközelítéshez* ("less is more" approach, LIMA) vezet a metaheurisztikus tervezésen belül.

Costa és szerzőtársai (2017) VNS heurisztikája a csere szomszédság (lásd a 6.3. ábrát) feltérképezésén alapul, amely tartalmazza az összes szomszédos megoldást. Hogy hatékonyan felderíthessék a csere szomszédságot, Huygen tétele alapján átfogalmazták a feladatot, így a célfüggvényt felírhatjuk a következő alakban:

$$\sum_{j=1}^{k} \frac{\sum_{i=1}^{n-1} \sum_{l=i+1}^{n} \|p_i - p_l\|^2 w_{ij} w_{lj}}{\sum_{i=1}^{n} w_{ij}}.$$

Mivel az egyenletes MSSC problémában a klaszterek számossága előre ismert, ezt felbonthatjuk két részre:

$$\frac{1}{\lceil \frac{n}{k} \rceil} \sum_{j \in \mu} \sum_{i=1}^{n-1} \sum_{l=i+1}^{n} \|p_i - p_l\|^2 w_{ij} w_{lj} + \frac{1}{\lfloor \frac{n}{k} \rfloor} \sum_{j \in \eta} \sum_{i=1}^{n-1} \sum_{l=i+1}^{n} \|p_i - p_l\|^2 w_{ij} w_{lj}, \tag{6.11}$$

ahol μ és η azon klaszterindexek halmazai, amelyek számosságai rendre $\lceil \frac{n}{k} \rceil$ és $\lfloor \frac{n}{k} \rfloor$. Ekkor egy j^* klaszter költségét a $w_{j^*} Q w_{j^*}^T$ kvadratikus kifejezés adja, ahol $w_{j^*} = (w_{1j^*}, \ldots, w_{nj^*})$ és $Q = [q_{ab}]$, $q_{ab} = \|p_a - p_b\|^2/2$. Ezután egy, az `LCW` algoritmusnál (lásd 6.2.2. fejezet) látott módszerhez hasonló eljárást javasolnak a lehetséges cserék költségváltoztatásainak frissítésére és a csereszomszédság feltérképezésére. A frissítési lépés $\mathcal{O}(n)$, míg a feltérképezés $\mathcal{O}(n^2)$ időben végezhető el.

Costa és szerzőtársai (2017) VNS módszerében az x megoldásból kiindulva az x' véletlen szomszédot folyamatosan bővülő $\mathcal{N}_t$ csere szomszédságból választják (de $\mathcal{N}_t \subset \mathcal{N}_{t+t_{step}}$ nem feltétlenül teljesül). Ennek megfelelően, ha $t = 2$, akkor x'-t abból a szomszédságból választják, amelyet az összes olyan megoldás alkot, amelyet x-ből kiindulva két véletlen cserével kaphatunk. Hasonlóan, ha $t = 3$, akkor x'-t három véletlen cserével kapjuk, és így tovább.

A lokális *leereszkedési módszer* (descent method) abból áll, hogy egy adott megoldásból kiértékelik azokat a szomszédokat, amelyeket különböző klaszterekben lévő pontok cseréjével kapnak, és az első javító szomszédra mozognak. Ez eltér a szokásos *gradiens leereszkedő módszerektől* (gradient descent method), amelyek a legmeredekebb, vagyis a célfüggvény értékének legnagyobb javulásával járó leereszkedő irányt választják a további

kereséshez. Hansen és Mladenović (2006) átfogó számítógépes kísérletekkel bemutatják, hogy amennyiben a kezdőmegoldást véletlenszerűen inicializáltuk, akkor az első javító irány választása javasolt.

A VNS-hez általában nagyon kevés paramétert kell beállítani. Costa és szerzőtársai (2017) VNS-LIMA heurisztikájában csupán két paraméter található: t_{max} és t_{step}. Ezeknek a kezdeti számítógépes kísérleteket követően a $t_{max} = \lfloor n/2 \rfloor$, illetve a $t_{step} = \lfloor t_{max}/20 \rfloor$ értékeket választották. A szerzők megjegyzik, hogy a kicsi t_{max} értékek nem voltak elegendőek ahhoz, hogy a VNS-LIMA kikerüljön a mély lokális optimumból. Továbbá, a kicsi t_{step} érték egy nagy t_{max} értékkel együtt tekintve rontja az algoritmus teljesítményét, mivel ekkor a szomszédságok kiterjesztése túl lassan történik. Ezzel ellentétben a nagy t_{step} érték nem hasznosítja azt a tényt, hogy a lokális minimumok jellemzően közel vannak egymáshoz a kombinatorikus optimalizálási problémáknál. A módszer pszeudokódját a 6.8. Algoritmus írja le.

Variable Neighborhood Search - Less Is More Approach (VNS-LIMA)

(Costa és szerzőtársai, 2017)

---

(1) $t_{max} = \lfloor n/2 \rfloor, t_{step} = \max\{1, \lfloor t_{max}/20 \rfloor\}$

(2) Egy tetszőleges $X = [x_{ip}]$ kezdőmegoldás megkonstruálása, ahol
$x_{ip} = 1$, ha $i = (p-1) * S + 1, (p-1) * S + 2, \ldots, (p-1) * S + S$ és
$p = 1, 2, \ldots, G$, valamint 0 minden más esetben.

(3) $R \leftarrow DX$, ahol $D = [d_{ij}]$ az elemek közötti négyzetes távolságok mátrixa.
cost $\leftarrow$ a megoldás költségének számítása (6.11) alapján.
$t \leftarrow 1$

(4) *Véletlenszerűen választunk egy elemet az $\mathcal{N}_t$ szomszédságból*
$X^{tmp} \leftarrow X, R^{tmp} \leftarrow R, \text{cost}^{tmp} \leftarrow \text{cost}$
for $tt = 1 : t$ [végrehajtunk t véletlenszerű cserét]
$g1, g2 \leftarrow$ véletlenszerűen választunk két csoportot
$l1, l2 \leftarrow$ véletlenszerűen választunk 1-1 elemet a két csoportból
$x^{tmp}_{l1,g1} \leftarrow 0, x^{tmp}_{l1,g2} \leftarrow 1, x^{tmp}_{l2,g2} \leftarrow 1, x^{tmp}_{l2,g1} \leftarrow 0$ [kicseréljük a két elemet]
$R^{tmp}_{\bullet g2} \leftarrow R^{tmp}_{\bullet g2} - D_{\bullet l2} + D_{\bullet l1}$
$R^{tmp}_{\bullet g1} \leftarrow R^{tmp}_{\bullet g1} - D_{\bullet l1} + D_{\bullet l2}$
end

(5) *Végrehajtjuk a lokális leereszkedés módszert*

$i \leftarrow 1, g_i \leftarrow$ az a csoport, amelybe az i elem tartozik

for $j = \{j \mid j \notin g_i\}$

$g_j \leftarrow$ az a csoport, amelybe a j elem tartozik

$\Delta_{ij}^{\text{cost}} \leftarrow R_{ig_j}^{tmp} - R_{ig_i}^{tmp} + R_{jg_i}^{tmp} - R_{jg_j}^{tmp} - 2D_{ij}$

if $\left(\Delta_{ij}^{\text{cost}} < 0\right)$ [ha javítanánk a cserével]

$x_{i,g_i}^{tmp} \leftarrow 0, x_{i,g_j}^{tmp} \leftarrow 1, x_{j,g_j}^{tmp} \leftarrow 1, x_{j,g_i}^{tmp} \leftarrow 0$ [kicseréljük a két elemet]

$R_{\bullet g_j}^{tmp} \leftarrow R_{\bullet g_j}^{tmp} - D_{\bullet j} + D_{\bullet i}$

$R_{\bullet g_i}^{tmp} \leftarrow R_{\bullet g_i}^{tmp} - D_{\bullet i} + D_{\bullet j}$

$\text{cost}^{tmp} \leftarrow \text{cost}^{tmp} + \Delta_{ij}^{\text{cost}}$

goto Step (5)

end if

end

(6) *Megnézzük, hogy tudtunk-e javítani az új lokális megoldással. Ha nem, és elértük a t maximális értékét, akkor az algoritmus terminál.*

if $(\text{cost}^{tmp} < \text{cost})$ [javítottunk]

$X \leftarrow X^{tmp}, R \leftarrow R^{tmp}, \text{cost} \leftarrow \text{cost}^{tmp}$

$t \leftarrow 1$

goto Step (4) [$\mathcal{N}_1$-ből újraindítjuk a keresést]

end if

$t \leftarrow t + t_{step}$

if $(t < t_{max})$

goto Step (4) [$\mathcal{N}_t$-ből újraindítjuk a keresést]

end if

6.8. Algoritmus. Változó Szomszédság Keresés - A Kevesebb Több Megközelítés (Variable Neighborhood Search - Less Is More Approach, VNS-LIMA) (Costa és szerzőtársai, 2017)

Algoritmusukat Malinen és Fränti (2014) egyenletes k-közép heurisztikájával hasonlítják össze, amelyet Costa és szerzőtársai (2017) az addigi legkorszerűbb eljárásnak nevez. Az elemzéshez való életből származó adatokat használnak fel.[9] Összesen 16 adathalmazt szerepeltetnek, köztük a Malinen és Fränti (2014) által alkalmazott 'Thyroid', 'Wine' és

[9] Az elemzésben felhasznált adatok letölthetőek a `http://archive.ics.uci.edu/ml/` címen.

'Iris' nevű adatokat is. Az elemek száma 150-2310-ig, a dimenziók száma 4-240-ig, a csoportok száma pedig 2-15-ig terjed.

A kísérletek alapján a VNS-LIMA módszer jobban teljesít az egyenletes k-közép eljárásnál, kevesebb idő alatt jobb eredményeket talál meg. A 7. Fejezetben elvégzett elemzésben a VNS-LIMA módszerre `Costaetal` névvel hivatkozunk.

6.2.7. Jitta-Klami Algoritmus (`JittaKlami`)

Jitta és Klami (2018) valószínűségszámításon alapuló klaszterezési modellek egy családját mutatja be. Megközelítésükben előzetesen megadható a lehetséges klaszterméretek eloszlása, így elősegítve a kívánt méretű csoportok létrejöttét. Előzetes eloszlásként megadható bármilyen multimodális eloszlás (olyan, amelynek több maximumhelye is lehet), amely így általánosítja a korábbi klaszterezési megközelítéseket, mint például az egyenlő méretű vagy a kismerétű klaszterek keresésének problémáit. A modell előnye, hogy egy rugalmas és finomhangolható eszközt ad az elemző kezébe a célnak (a modell keretein belül) leginkább megfelelő klaszterek kialakításához.

A klaszterezési modellüket valószínűségszámítási nézőpontból írják fel, ahol az elemek adott véges $X = \{x_n\}_{n=1}^{N}$ halmazát D-dimenziós vektorok reprezentálják, és az elemek valamilyen, azokat generáló eloszlásokból származnak (*generative models*). A modelleket két tényező határozza meg: 1) N darab z_n látens változó, amelyek azt mutatják, hogy az egyes elemeket melyik csoportba soroljuk, valamint 2) K eloszlás, amelyek a klaszterallokációra feltételesen generálják a megfigyeléseket. A klaszterallokációkat a θ_k paraméterek adják meg.

A modellcsalád egy tipikus példája McLachlan és Peel (2000) véges kevert modellje, ahol a minta együttes eloszlását

$$p(X, Z|\theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} [p(z_n = k)p(x_n|\theta_k)]^{\mathbb{I}_{[z_n=k]}}$$

írja le, ahol $\mathbb{I}_{[.]}$ az identitásfüggvény, melynek értéke 1, ha a szögletes zárójelek között lévő kifejezés logikai értéke igaz, ellenkező esetben pedig 0.

A gyakorlat szempontjából fontos részlet, hogy ebben az esetben a paraméterek becslésénél minden egyes mintát függetlenül lehet tekinteni, mivel $p(Z|X, \theta) = \prod_{n=1}^{N} p(z_n|x_n, \theta)$, és így a probléma hatékonyan megoldható.

Jitta és Klami (2018) az általuk javasolt kevert-alapú klaszterezésben, a klasztermére-

tek direkt befolyásolását is lehetővé teszi. A független, azonos eloszlású (*i.i.d.*) mintákat független, azonos eloszlású klaszterekre cserélik fel:

$$p(X, Z|\theta) = \prod_{k=1}^{K} \left[p(s_k) \prod_{n=1}^{N} p(x_n|\theta_k)^{\mathbb{I}[z_n=k]} \right], \tag{6.12}$$

ahol s_k jelöli a k-adik klaszterben lévő mintaelemek számát. Ezzel a felírással, egy megfelelő $p(s)$ eloszlás megadása révén, meghatározhatunk egy előzetes preferenciát a klaszterméretekre. Ugyanakkor ezzel együtt el is veszítjük azt az előnyt, hogy a klaszterallokációkat minden mintára egymástól függetlenül határozhassuk meg.

A (6.12)-es formula által leírt együttes eloszlás maximalizálására egy alternáló algoritmust használhatunk. Egy kezdeti θ_k választás mellett az alábbi két lépést váltogatjuk:

1. $p(Z|X, \theta)$ alapján megbecsüljük az új Z allokációt együttesen minden mintára.
2. Új θ_k paramétereket becslünk $p(\theta_k|X_k)$ alapján minden $k = 1, \ldots, K$ klaszterre, ahol X_k azon elemek halmaza, amelyekre $z_n = k$.

A tanulmányukban Jitta és Klami (2018) a *maximum a posteriori* megoldást keresik, amely esetén így az első lépés egy egyszerű korlátozott optimalizálási feladatnak felel meg. A második lépés a választott likelihoodtól függ, és a hagyományos *kevert modellezéssel* (mixture modeling) azonos.

Az első lépés

Az első lépésben a hozzárendelést az alábbi együttes log-likelihood maximalizálásával határozzák meg:

$$\begin{aligned} \log p(Z|X, \theta) &= \log p(X|\theta, Z) + \log p(Z) \\ &= \sum_i \log p(x_i|\theta_{z_i}) + \sum_j \log p(s_j), \end{aligned} \tag{6.13}$$

amelyet minden z_n-re egyszerre kell megoldani, mivel s_j jelöli a j-edik klaszterhez rendelt elemek számát.

Jelölje

$$C_{ij} = \log p(x_i|\theta_j)$$

az i-edik adatpont log-likelihoodját a j-edik klaszterre, vagyis annak a log-valószínűségét, hogy az i-edik adatpont a j-edik csoportba tartozik. Legyen továbbá $\mathcal{S} = \{S_1, \ldots, S_A\}$ a $p(s)$ valószínűség tartója, ahol A jelöli a lehetséges, különböző klaszterméretek számát,

valamint S_1 adja a legkisebb, nem nulla valószínűségű, lehetséges klaszterméretet, míg S_A a legnagyobb, nem nulla valószínűségű klaszterméretet. Ekkor jelölje

$$d_a = \log p(s = S_a)$$

az S_a lehetséges klaszterméret log-priorját.

A szükséges inputok bármely olyan módszerhez, amely alkalmas a (6.13)-es képlet maximalizálására, összegyűjthetőek egy $C = [C_{ij}]$ mátrixban és egy $d = [d_a]_{a=1,\dots,A}$ vektorban.

Jitta és Klami (2018) két gyakorlati módszert javasol, amelyek korlátozott optimalizálást felhasználva, a log-likelihood maximalizálásával oldják meg a hozzárendelési feladatot. Az első egy *bináris egészértékű programozási feladatként* (binary integer program) tekint a problémára, míg a második egy *kevert egészértékű programozási feladatként* (MILP). A 7. Fejezetben bemutatott elemzés során az elsőt alkalmazzuk, így itt ezt részletezzük.

Az együttes hozzárendelési feladat tetszőleges előzetes eloszlásokra megoldható, ha azt bináris egészértékű programozási feladatként írjuk fel. Ezt egy $\Pi \in [0,1]^{N\times K}$ mátrix és egy $B \in [0,1]^{K\times A}$ mátrix parametrizálja. $\Pi_{ij} = 1$ jelöli, hogy az i-edik mintaelemet a j-edik klaszterhez rendeljük, és $B_{ja} = 1$ felel meg annak, hogy a j-edik klaszternek pontosan S_a eleme van. Vegyük észre, hogy a Π és B mátrixok minden egyes sorában pontosan egy nem nulla elem lehet (*special ordered set of type 1 - SOS1* típusú mátrixok).

A bináris egészértékű programozási feladat a következőképpen írható fel:

$$\begin{aligned}
&\max \sum_{i=1}^{N}\sum_{j=1}^{K} C_{ij} \times \Pi_{ij} + \sum_{j=1}^{K}\sum_{a=1}^{A} B_{j,a} \times d_a \qquad (6.14)\\
&\text{s.t.} \sum_{j=1}^{K} \Pi_{ij} = 1 \quad \forall i,\\
&\sum_{a=1}^{A} B_{j,a} = 1 \quad \forall j,\\
&\sum_{i=1}^{N} \Pi_{ij} = \sum_{a=1}^{A} B_{j,a} \times S_a \quad \forall j,
\end{aligned}$$

ahol az $\times$ művelet elemenkénti szorzást jelöl. Ebben $(N + A)K$ bináris változó szerepel, és $N + 2K$ feltételt használunk, amelyek közül $N + K$ SOS1 típusú.

A második lépés

Ha minden klaszter azonos méretű kell, hogy legyen, akkor $\mathcal{S} = \{S_1\}$ egyelemű halmaz, és $d_a = \log p(s = S_1) = 0$, így a fenti (6.14)-es bináris egészértékű programozási feladat az alábbi szerint egyszerűsödik:

$$\begin{aligned} \max & \sum_{i=1}^{N} \sum_{j=1}^{K} C_{ij} \times \Pi_{ij} \\ \text{s.t.} & \sum_{j=1}^{K} \Pi_{ij} = 1 \quad \forall i, \\ & \sum_{i=1}^{N} \Pi_{ij} = k \quad \forall j. \end{aligned} \tag{6.15}$$

A második lépésben *maximum a posteriori (MAP)* becslést alkalmazunk. Legyen adva egy θ megbecsülendő paraméter és egy $\mathcal{D} = \left\{x^{(1)}, \ldots, x^{(N)}\right\}$ adathalmaz. Legyen továbbá $p(\theta|\mathcal{D})$ a posterior eloszlás. Ekkor a MAP becslés azt a $\hat{\theta}$ paramétert választja, amelyik a legvalószínűbb a posterior eloszlás mellett, vagyis amelyik maximalizálja azt:

$$\begin{aligned} \hat{\theta}^{MAP} &= \arg\max_{\theta} p(\theta|\mathcal{D}) \\ &= \arg\max_{\theta} p(\theta, \mathcal{D}) \\ &= \arg\max_{\theta} p(\theta) p(\mathcal{D}|\theta) \\ &= \arg\max_{\theta} \left[\log p(\theta) + \log p(\mathcal{D}|\theta)\right]. \end{aligned}$$

Ha az adatok eloszlására normális eloszlást feltételezünk ismeretlen μ várható értékkel és ismert σ szórással, és a prior eloszlásra a konjugált prior eloszlást, vagyis szintén normális eloszlást feltételezünk, akkor a posterior eloszlás szintén normális lesz. A MAP paraméterbecslés eredménye ekkor

$$\hat{\mu}^{MAP} = \frac{\frac{1}{(\sigma^{pri})^2}\mu_{pri} + \frac{N}{\sigma^2}\frac{1}{N}\sum_{i=1}^{N} x^{(i)}}{\frac{1}{(\sigma^{pri})^2} + \frac{N}{\sigma^2}}, \tag{6.16}$$

ahol μ^{pri} és σ^{pri} a prior normális eloszlás várható érték és szórás paramétereit jelölik.

Finomhangolás

A hagyományos, valószínűségszámításon alapuló klaszterezési modellek a klaszterek számának megadását és mellette esetleg azok relatív súlyának specifikálását követelik meg. Jitta és Klami (2018) felírásában ezeket két új lehetőség váltja fel, amelyekkel a klaszterek

mérete és száma befolyásolható.

A nagyobb jelentőséggel bíró rész a likelihood és a prior eloszlás relatív szerepe. Ha a méretpreferenciákat egy közel lapos prior adja, akkor a célfüggvényben a likelihood rész a meghatározó, és a prior egyetlen hatása azon megoldások kizárása, amelyekhez a $p(s)$ prior nulla valószínűséget rendel. A másik extrém eset az, amikor a prior dominál a célfüggvényben, és csak olyan megoldások jöhetnek szóba, amelyek pontosan a kívánt méretű klasztereket adják. Jitta és Klami (2018) a gyakorlatban a prior növekvő súlyozásával javasolja az algoritmus futtatását egészen addig, amíg a klaszterek empirikus hisztogramja kellőképpen megfelel a prior eloszlásnak.

A másik lehetőség $p(s = 0)$ megadása, vagyis az üres klaszterek valószínűségének meghatározása, amely a klaszterek K számának megválasztásával együtt befolyásolja az adatok modellezéséhez használt klaszterek tényleges számát. Egy gyakorlatban használható stratégia, hogy először megbecsüljük a klaszterek várható számát a prior eloszlás alatt, amely $\hat{K} = \frac{N}{\mathbb{E}_{p(s)}[s]}$, ahol $\mathbb{E}_{p(s)}[\cdot]$ jelöli a $p(s)$ eloszlás és $s > 0$ esetén a várható értéket. Ezután úgy futtatjuk a modellt, hogy a K értékét kicsivel magasabbra állítjuk, mint $\hat{K}$, például $K = 1.5\hat{K}$-ra, és hagyjuk, hogy a modell automatikusan kihagyja azokat a klasztereket, amelyek nem szükségesek az adatok modellezéséhez. A klaszterek kihagyásának mértékét a $p(s = 0)$ paraméter szabályozza.

Kísérletek

Az algoritmus inicializálásához valamilyen elfogadható θ_k klaszterparaméterek megadása szükséges, mielőtt még legelőször elvégeznénk a hozzárendeléseket. A konvergencia meggyorsítása érdekében érdemes a kezdő klasztereket egyenletesen eloszlatni az adatok által meghatározott térben. Jitta és Klami (2018) a k-közép++ algoritmus által használt inicializálási stratégiát választották, így a későbbi elemzés során mi is ezt alkalmazzuk.

A szerzők által alkalmazott modell Gauss klasztereket használ $\Sigma_j = \sigma^2 \mathbb{1}$ izotropikus kovarianciával és azonos prior eloszlásokat feltételeznek a klaszterközéppontokra. A klaszstermodellek javasolt családján belül ez a modell áll a legközelebb a k-közép klaszterezéshez, és így σ^2 kontrollálja a klaszterméretekre adott prior információk figyelembevételét. Ezt könnyedén láthatjuk, hogyha a

$$\log p(X|Z, \mu) + \log p(Z) = -\frac{1}{2\sigma^2} \sum_n \|x_n - \mu_{z_n}\|^2 + \sum_j \log p(s_j)$$

célfüggvény $2\sigma^2$-szeresét vesszük, és így az ezzel ekvivalens

$$-\sum_n \|x_n - \mu_{z_n}\|^2 + 2\sigma^2 \sum_j \log p(s_j) \tag{6.17}$$

célfüggvényhez érkezünk.

Jitta és Klami (2018) módszerének implementációját a 6.9. Algoritmus szerinti pszeudokód írja le. Mivel a $\theta_j, j = 1, \ldots, K$ paraméterek a normális eloszlás várható érték paramétereinek felelnek meg, ezért a leírásban μ_j és $\mu = \{\mu_1, \ldots, \mu_K\}$ szerepel θ_j valamint θ változók helyett.

Jitta-Klami (JK) Algoritmus (Jitta és Klami, 2018)

---

(1) *Meghatározzuk véletlenszerűen a kezdő μ_j klaszterparamétereket.*

step $\leftarrow 0$

$\{\mu_j\}_{j=1,\ldots,K} \leftarrow$ kezdő klaszterekközéppontok a k-közép++ algoritmus inicializálása szerint

(2) *Új Z allokációk becslése együttesen a minta egészére $p(Z|X, \mu)$ alapján.*

$\Pi \leftarrow$ a (6.15)-es bináris egészértékű programozási feladat megoldásával

$Z = \{z_1, \ldots, z_N\} \leftarrow \Pi \times [1, \ldots, K]^T$ [új klaszterhozzárendelések]

if (step $<$ max_step **and** Z megváltozott)

goto Step (3)

else

stop

end if

(3) *Új μ_j paraméterek becslése minden klaszterre $p(\mu_j|X_j)$ alapján, ahol X_j azon elemek halmaza, amelyekre $z_n = j$.*

$\hat{\mu}^{MAP} \leftarrow$ (6.16) alapján, ahol $\hat{\mu}^{MAP} = \{\hat{\mu}_1^{MAP}, \ldots, \hat{\mu}_K^{MAP}\}$ a klaszterenként újrabecsült paraméterek halmaza

goto Step (2)

6.9. Algoritmus. Jitta-Klami (JK) Algoritmus (Jitta és Klami, 2018)

6.3. Összefoglaló táblázat

A fejezet során bemutatott algoritmusokat három csoportba soroljuk, és a módszereket a 6.10. Táblázatban összegezzük. A konstruktív módszerek egy nem megengedett megoldásból indulnak ki, és a klaszterek 'felépítése' a távolságok alapján csoportok összevonásával (`cluster`) vagy elemek áthelyezésével (`eq1-6`) történik. A középpontok számítását és hozzárendelését alternáló eljárások közé azon módszereket soroljuk, amelyek lényegi részét ez a két lépés képezi. A lokális keresést alkalmazó heurisztikák megengedett megoldásokból indulnak ki, és alapvetően az elemeken keresztül iterálva, lokális keresés révén próbálnak javítani a célfüggvény értékén.

Konstruktív módszerek

`cluster`	Összevonó hierarchikus klaszterezés alkalmas vágással, amely távolságmértékként négyzetes euklideszi távolságokat, összevonó eljárásként pedig Ward módszert alkalmaz
`eq1-6`	Heurisztikus eljárások, amelyek az elemek és a klaszterközéppontok távolságai alapján, lépésenként egy elem áthelyezésével egyenlítik ki a csoportokat

Középpontok számítását és hozzárendelést alternáló eljárások

`kmeans`	A k-közép++ módszer, amely a klaszterközéppontok számítását és a pontoknak a hozzájuk legközelebb lévő klaszterhez történő hozzárendelését alternálja
`eqFCM`	A fuzzy c-közép egyenlő klaszterméretekkel módszer, amely prototípusok és hozzátartozási mértékek számítását alternálja
`eqFCMv2`	Az `eqFCM` módszer futása után elvégezzük a csoportok kiegyenlítését úgy, hogy a hozzátartozási mértékek szerint csökkenő sorrendben haladva rendeljük hozzá az elemeket klaszterekhez
`MalinenFranti`	A `kmeans` eljáráson alapuló alternáló módszer, ahol az elemek hozzárendelése a klaszterekhez a Magyar módszer segítségével történik

JittaKlami	Valószínűségszámításon alapuló módszer, amely az elemek allokációinak és a klaszterek paramétereinek becsléseit alternálja

Lokális keresést alkalmazó heurisztikák

LCW	Az eljárás párok cseréjével próbál meg javítani a célfüggvényértéken
LCWv2-4	A módszerek párok és hármas cserék segítségével próbálnak javítani a célfüggvényértéken különböző konstrukciók szerint
TLCW	Az LCW módszer mohó konstrukcióval és tabu memóriával, ahol egy lokális megoldás megtalálása után végrehajtjuk a legjobb nem javító cserét, és újraindítjuk a keresést
SO	A TLCW módszer stratégiai ingázással ötvözve, amely a keresés során nem megengedett megoldásokat is érint
Costaetal	A LIMA-VNS módszer alkalmazása, amely növekvő szomszédságokból egy pontot véletlen cserék útján kiválasztva indítja újra az LCW eljárást

6.10. Táblázat. Az elemzésben szereplő heurisztikus módszerek összefoglaló táblázata.

7. fejezet

Elemzés

A következőekben bemutatjuk a 6. Fejezetben tárgyalt algoritmusok elemzését. Az eljárások összevetése során egyes módszerek minimalizálási és maximalizálási verzióját is szerepeltetjük, ha az átírás magától értetődő. Így például az `LCW` algoritmus esetében tekintünk `LCWmin` és `LCWmax` algoritmusokat is rendre a minimalizálási, valamint a maximalizálási problémákra. Hasonlóan a hármas cseréket megvalósító eljárásokra, továbbá a `TLCW`, az `SO` és `Costaetal` algoritmusokra is. A `MalinenFranti` algoritmus átírását is megkíséreltük, de az előzetes futtatások során nem adott jó eredményeket, így végül nem szerepeltetjük az elemzésben. Továbbá amely heurisztikában eredetileg egyszerű távolságok jelentek meg, ott az elemzéshez távolságnégyzeteket alkalmazunk.

A fejezet során a célunk, hogy betekintést nyerjünk a megfogalmazott m-szobatárs probléma tulajdonságaiba és a gyakorlati megoldhatóságába. Amellett, hogy vizsgáljuk, hogy mely módszerek teljesítenek a legjobban, külön figyelemmel kísérjük a hármas cseréket is alkalmazó módszereket. Ezek esetében azt is megnézzük, hogy más heurisztikák eredményeit, mint kezdőmegoldást alkalmazva, tudunk-e javítani az eredeti célfüggvényértéken.

Megjegyezzük, hogy a kísérletekhez a hallgatói minták generálásánál általánosan a tulajdonságok száma 3, és minden tulajdonság a $[0, 10]$ intervallumból vehet fel egész értéket. Az elemzés elvégzéséhez szükséges programokat MATLAB-ban implementáltuk, az algoritmusok futtatását pedig egy Intel Core i5-8600K 3.60GHz processzorral és 16,0 GB RAM-mal rendelkező számítógépen hajtottuk végre.

7.1. A kiegyenlítő eljárások vizsgálata

A 6.1.4. Fejezetben bemutatott kiegyenlítő módszerek összehasonlítása Monte-Carlo (MC) szimulációval történik. Az összehasonlítást minden esetben valamelyik rögzített alap klaszterező eljárás mellett tesszük meg, amely az összevonó hierarchikus klaszterezés alkalmas vágással (`cluster`), a k-közép++ (`kmeans`), vagy a fuzzy c-közép (`eqFCM`) heurisztikák valamelyike.

Egy szcenárió esetén véletlenül generált hallgatókat tekintünk, alkalmazzuk az alap klaszterező eljárást, majd az így kapott csoportosításra futtatjuk valamennyi kiegyenlítő módszert (`eq1-eq6`), amelyekről meghatározzuk, hogy egymáshoz képest mennyire teljesítettek jól. Ezután a módszerek összehasonlítása különböző mutatók mentén történik. Egy MC szimuláció során 10000 mintát generálunk, így a mutatók az egyes szcenáriók aggregált eredményeit tükrözik. A mutatók stabilitásának vizsgálatához ezt a folyamatot 100-szor megismételjük, és meghatározzuk a 100 futás alapján a mutatók különböző statisztikáit is: a mediánt, a minimum és maximum értékeket, valamint a 25%-os, illetve 75%-os percentiliseket.

Ahhoz, hogy egy szcenárió esetén meghatározzuk, hogy az egyes kiegyenlítő eljárások relatíve mennyivel teljesítenek jobban vetélytársaiknál, a következő értékelést alkalmazzuk. Legyen $c_1, c_2, \ldots, c_6$ rendre az `eq1`, `eq2`, ..., `eq6` kiegyenlítő eljárás által kapott költség, és jelölje

$$c_{terj} = \max\{c_1, \ldots, c_6\} - \min\{c_1, \ldots, c_6\}$$

a költségek terjedelmét. Ekkor a

$$p_i = \begin{cases} \sum_{\substack{1 \le j \le 6 \\ j \neq i}} \frac{c_j - c_i}{c_{terj}}, & \text{ha } c_{terj} > 0, \\ 0, & \text{egyébként} \end{cases}$$

relatív pontszám mutatja, hogy az i-edik kiegyenlítő eljárás relatíve mennyivel teljesít jobban a többinél. A terjedelemmel való skálázásra azért van szükség, mert a generált hallgatóktól függően a minimum és maximum költségek közötti távolság elég változékony lehet. A skálázás révén a legjobban és legrosszabbul teljesítő módszerek skálázott költségei közötti távolság mindig 1, a többi eljárás skálázott költségkülönbségei pedig ezzel arányosan adódnak.

Az algoritmusok értékelésre a következő mutatókat vezetjük be:

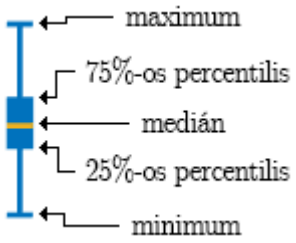
p0 : Azt mutatja, hogy az algoritmus hányszor adta vissza a legjobb eredményt (döntetlenek is számítanak), osztva az ismétlések számával.

p1 : 6-tól 1-ig pontozzuk az eljárásokat, és ezek átlagát vesszük algoritmusonként. A legjobb eljárás 6 pontot kap, a legrosszabb pedig egyet. Döntetlenek esetén az algoritmusok egységesen a helyezésekért járó pontok átlagát kapják. Így ha a két legjobban teljesítő algoritmus ugyanolyan jól szerepelt, akkor mindkettő 5,5 pontot kap. Ha az élen hármas döntetlen alakult ki, akkor mindhárom eljárás 5 pontot kap. A pontszámok egy lehetséges alakulása rendre az `eq1`, ..., `eq6` módszerekre: (legjobb) $[5{,}5; 5{,}5; 4; 2; 2; 2]$ (legrosszabb).

p2 : Átlagosan mennyire teljesít jól az algoritmus a többi módszerhez képest, vagyis a p_i relatív pontszámok átlaga.

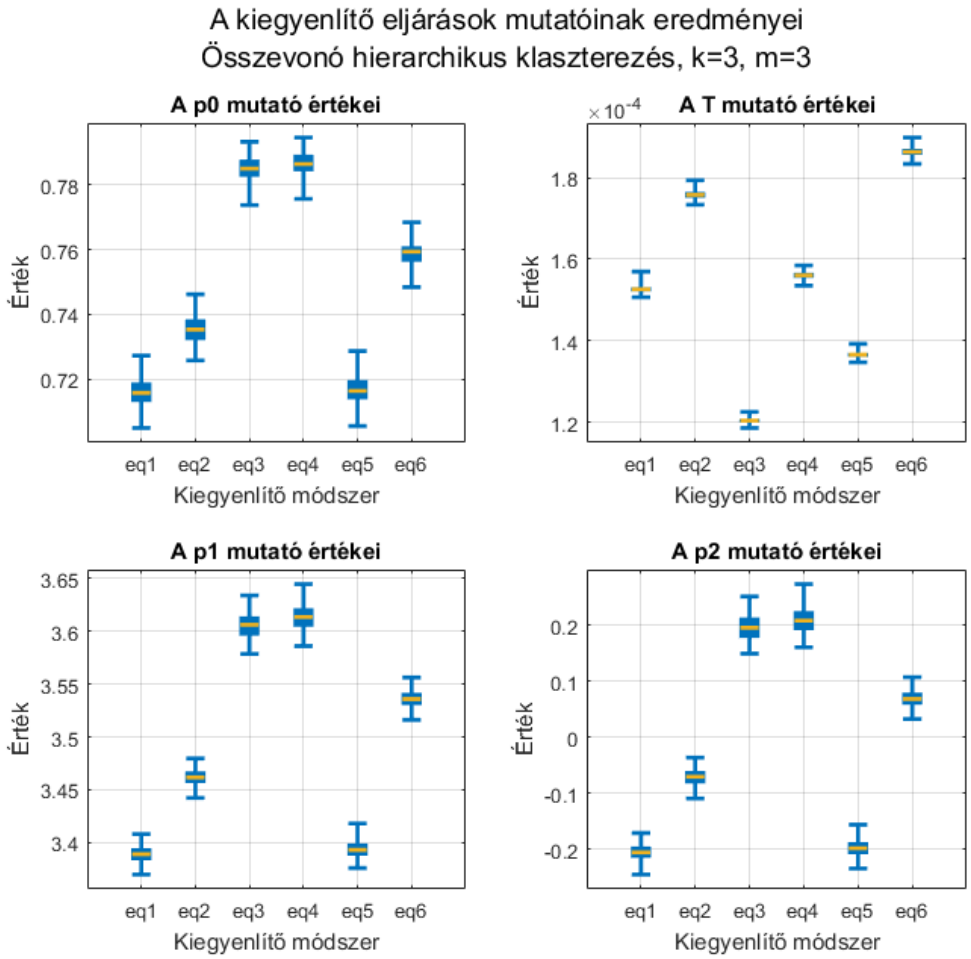
T : Átlagos futásidők.

Az algoritmusok összehasonlításához használt ábrákon a 100 futtatás eredményéből számított statisztikák megjelenítésére gyertyaszerű ábrákat alkalmazunk. Ezek a 7.1. ábrán jelölt módon mutatják a maximum (felső vízszintes kék vonal), a 75%-os percentilis (a középső széles terület felső éle), a medián (a középső vízszintes sárga vonal), a 25%-os percentilis (a középső széles terület alsó éle) és a minimum (az alsó vízszintes kék vonal) értékét.



7.1. ábra. A kiegyenlítő eljárások statisztikai mutatóinak megjelenítésére használt gyertyaszerű ábrák magyarázata.

A kiegyenlítő eljárások eredményeit $k = 3$ és $m = 3$ esetére a 7.2., 7.3. és 7.4. ábrák mutatják be. A következő bekezésekben ezeket értékeljük ki. Ehhez hasonlóan mutatják be az eredményeket $k = 5$ és $m = 3$ esetében a Melléklet A.2. fejezetének A.2., A.3. és A.4. ábrái. Az ezekhez kapcsolódó értékeléseket a vonatkozó részeknél lábjegyzetben adjuk meg.



7.2. ábra. A kiegyenlítő eljárások (`eq1-eq6`) mutatóinak eredményei. Összevonó hierarchikus klaszterezés, $k = 3, m = 3$.

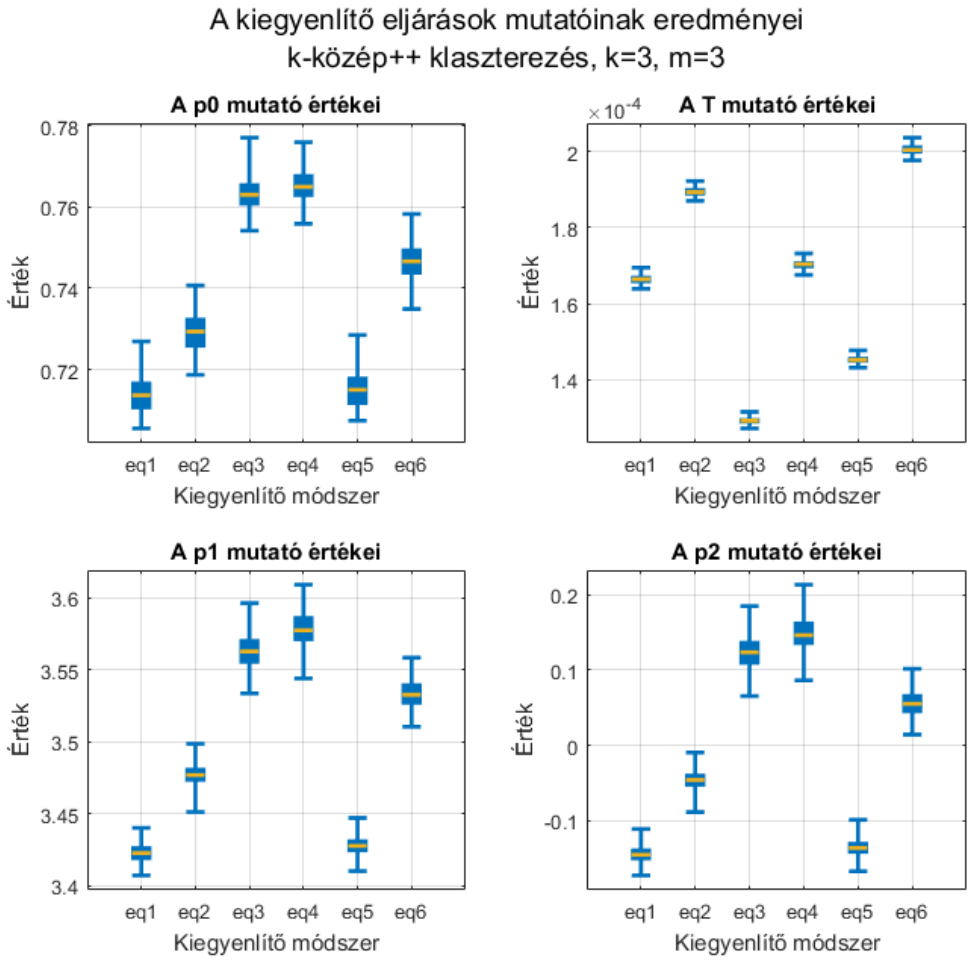
A $k = 3$ és $m = 3$ paraméterekre a futásidő (T mutató) szempontjából minden esetben az `eq3` heurisztika teljesített a legjobban. Ezt az `eq5` követi nem sokkal lemaradva, majd az `eq1` és az `eq4` következik. A legrosszabb futásidőket minden esetben az `eq2` és az `eq6` módszerek produkálták. A gyertyák magassága alapján a sorrend stabil.

Aszerint, hogy melyik algoritmus az esetek hány százalékában adta a legjobb eredményt (p0 mutató), valamennyi esetben az `eq3` és `eq4` módszerek teljesítettek a legjobban, és ezeket követi nagyobb lemaradással az `eq6` eljárás.[1] Megjegyezzük, hogy az `eqFCM` eljárásra a gyertyák közötti átfedések miatt a sorrend esetenként eltérő lehet.

A p1, illetve p2 mutatók rendkívül nagy hasonlóságot mutatnak a p0 mutatóval, így ezek szerint is egyértelmű az értékelés. Minden esetben az `eq3` és `eq4` algoritmusok telje-

[1] $k = 5$ és $m = 3$ esetén az `eqFCM` eljárásra sokkal nagyobbak az átfedések, de általánosan továbbra is az `eq4` ejárás tűnik a legjobbnak, amelyet az `eq3` módszer követ.

sítenek a legjobban.[2]

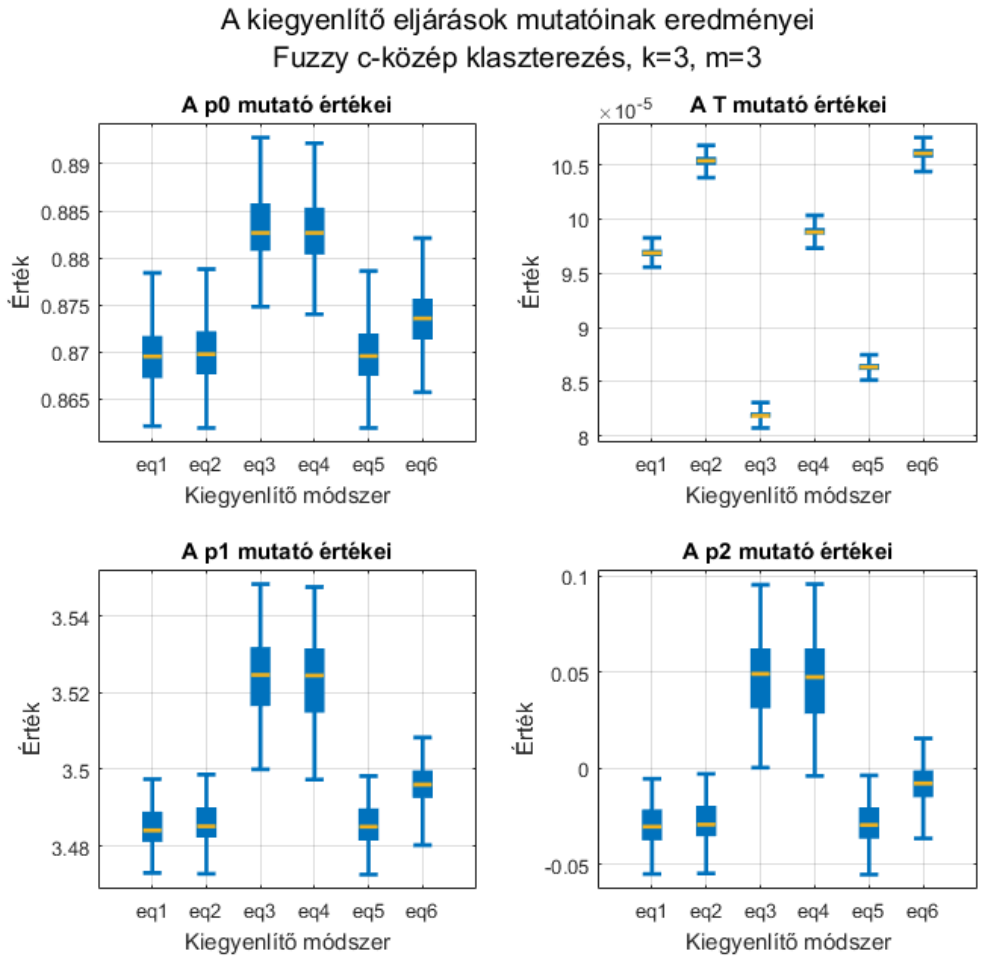


7.3. ábra. A kiegyenlítő eljárások (`eq1-eq6`) mutatóinak eredményei. k-közép++ klaszterezés, $k = 3, m = 3$.

Az elemzést $d = 10$ dimenzióra is elvégeztük, ahol többnyire hasonló eredményeket kaptunk. Az `eq3` és `eq4` előnye sokkal jelentősebb a többi eljáráshoz képest, az `eq4` futásideje pedig úgy tűnik, hogy javult, bár általánosságban továbbra sem előzte le az `eq5` eljárást.

Az elemzés eredményeképpen összességében az `eq4` módszert értékeljük a legjobbnak, amely futásidejét tekintve közepesen, a többi mutató szerint viszont általánosságban nagyon jól teljesített. Emellett a további kísérletek során rendkívül kedvező futásideje miatt az `eq3` módszert is szerepeltetjük.

[2] $k = 5$ és $m = 3$ esetén a p1 és p2 mutatók alapján a `cluster` és `kmeans` módszerekre az `eq3` egyértelműen lemarad az `eq4` kiegyenlítő eljárástól. A `kmeans` módszerre az `eq3` kissé elmarad már az `eq5` teljesítményétől is. Az `eqFCM` heurisztika esetében a p1 és p2 mutató szerint is a p0-tól eltérő sorrend rajzolódik ki. A p1 mutató szerint továbbra is az `eq4` módszer teljesít a legjobban, a p2 mutató esetében ugyanakkor látszólag az `eq2` módszer veszi át a vezető szerepet, habár az eredményekben rendkívül nagyok az átfedések.



7.4. ábra. A kiegyenlítő eljárások (`eq1-eq6`) mutatóinak eredményei. Fuzzy c-közép klaszterezés, $k = 3, m = 3$.

7.2. Kísérletek valós adatokon

A heurisztikus eljárások összehasonlításának első lépéseként az irodalomban is használt 'Iris' és 'Seeds' adathalmazokat tekintjük (lásd Malinen és Fränti (2014); Costa és szerzőtársai (2017); Rujeerapaiboon és szerzőtársai (2019)). Ezek olyan valós adathalmazok, amelyek egyenlő elemszámú klasztereket tartalmaznak. A megjelölt forrásokban ezeket a minimalizálási problémák esetén alkalmazták, és ismert rájuk a minimális célfüggvényérték. Ebből kifolyólag a fejezet során mi is a minimumkereső algoritmusok összehasonlításával foglalkozunk.

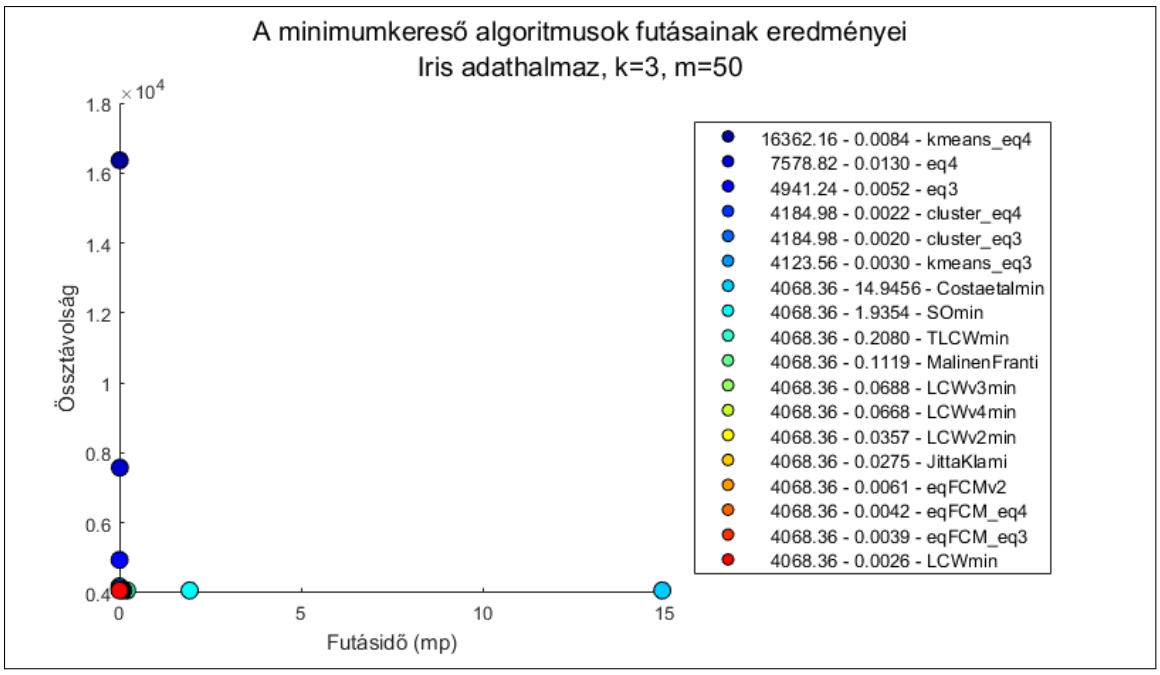
Az elemzéshez az algoritmusokat futtatjuk az adatokon, mérjük a futásidejüket, majd kiszámítjuk a megtalált megoldások célfüggvényértékét. Az eredményeket az össztávolságok és célfüggvényértékek terében a 7.5. és 7.6. ábrák mutatják be, valamint ezeket az

algoritmusok feliratain is feltüntetjük.

Az eredmények feliratai elsősorban a megtalált célfüggvényérték és másodsorban a futásidő szerint vannak sorbarendezve. Látható, hogy a heurisztikák többsége ugyanazt a minimumértéket találja meg, amely egyben az irodalom alapján a tényleges minimumnak gondolt érték.[3] A minimumértéket megtaláló eljárások között a futásidőkben elég nagy eltérések mutatkoznak, amely egyértelműen az algoritmusok konstrukciójából adódik. Amelyik módszer komplexebb módon próbál meg javítani a célfüggvényértéken, annak a futásideje óhatatlanul nagyobb lesz. Ezzel együtt viszont általános esetben várhatóan jobban teljesítenek a megadott megoldás tekintetében, mint az egyszerűbb társaik.

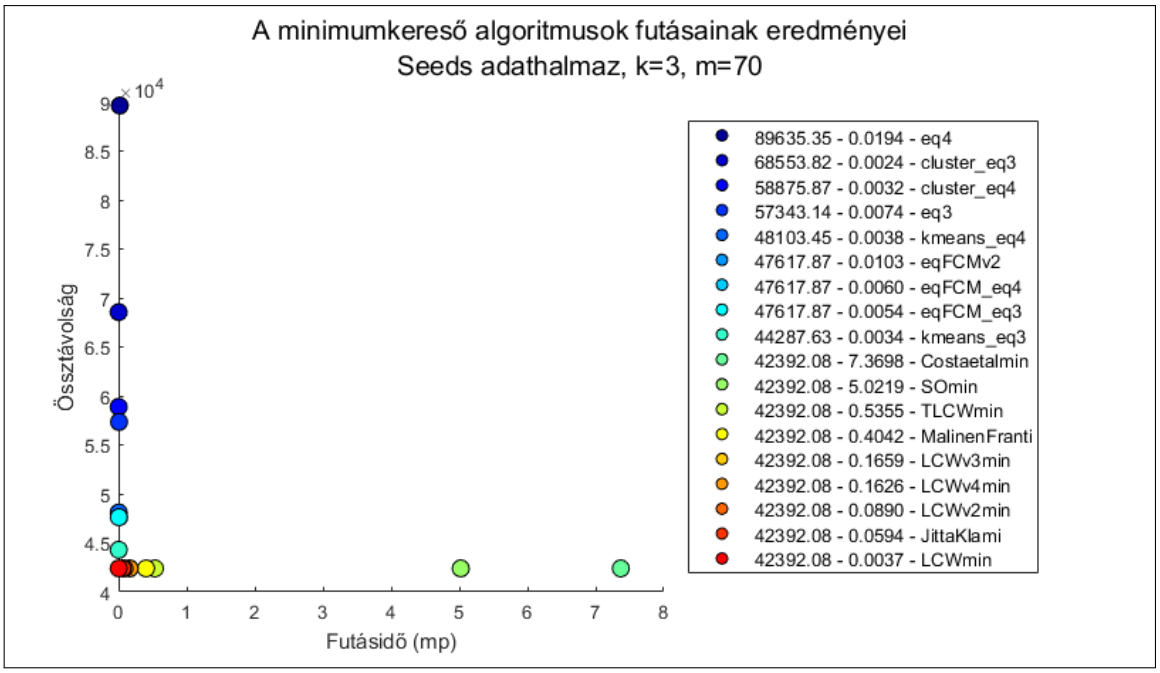
A hagyományos klaszterelemzéshez kapcsolódó módszerek, habár futásidő szempontjából jól teljesítenek, általánosan rosszabb - egyik esetben több, mint 4-szer rosszabb - célfüggvényértékeket találnak meg. Az alacsony futásidő kedvező lehet az adatpontok számának növekedésével, ugyanakkor a gyenge teljesítmény nem túl biztató.

Ahhoz, hogy jobban szemügyre tudjuk venni a heurisztikákat, szimulált adatokon további, átfogó elemzésnek vetjük alá őket.



7.5. ábra. A minimumkereső algoritmusok futásainak eredményei az Iris adathalmaz esetében. $k = 3, m = 50$. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

[3] A maximumkereső eljárások egyszeri futtatásával talált maximumértékek az Iris adathalmaz esetén 34041,12, míg a Seeds adathalmaz esetén 190389,59.



7.6. ábra. A minimumkereső algoritmusok futásainak eredményei a Seeds adathalmaz esetében. $k = 3, m = 70$. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

7.3. Kis k és kis m

Az elemzés második lépéseként kis k és m értékek esetén, szimulált adathalmazokon hasonlítjuk össze a különböző módszereket. Ekkor az összes lehetséges beosztást végignézve relatíve gyorsan meg tudjuk határozni az elérhető legnagyobb, illetve legkisebb költségeket, amelyekhez képest ki tudjuk értékelni a heurisztikákat. Az elemzésben a véletlenszerűen előállított adathalmazok dimenziója $d = 3$, vagyis a szimulált hallgatók 3 tulajdonsággal rendelkeznek. Az eljárások kiértékeléséhez meghatározzuk az összes lehetséges szobabeosztást, azok költségeit, az ezek által megadott eloszlást hisztogram formájában, valamint a tényleges optimumokat is. Ehhez a következő fejezetben leírt algoritmust alkalmazzuk.

7.3.1. A lehetséges szobabeosztások

Az összes lehetséges szobabeosztás felírására számokat rendelünk a hallgatókhoz 1-től n-ig. Ezután egy rekurzív eljárás segítségével lépésenként alakítjuk ki a szobákat. A rekurzív algoritmus pszeudokódjának megadásához az alábbi definíciókat és jelöléseket vezetjük be.

7.3.1. Definíció. *Jelölje $k > 1$ a szobák számát, $m > 1$ a szobák (azonos) méretét és $n = k \cdot m$ a hallgatók számát. Legyen $S = \{1, \dots, n\}$ az összes hallgató halmaza. Legyen továbbá $R \subset S, |R| = m$ egy beosztott szoba és $\mathcal{R} = \{R | R \subset S, |R| = m\}$ egy szoba lehetséges beosztásainak halmaza. Legyen megadott k és m értékek mellett az r-szobabeosztások $(0 \leq r \leq k)$ halmaza*

$$\begin{aligned}\mathcal{P}_r = \Big\{ & (R_1, \dots, R_r, s_1, \dots, s_l) \mid \\ & 0 \leq r \leq k;\ \emptyset \neq R_i \in \mathcal{R}\ \forall i;\ R_i \cap R_j = \emptyset\ \forall i, j, i \neq j; \\ & 0 \leq l \leq n;\ l = n - r \cdot m;\ s_i \in S\ \forall i;\ s_i \notin R_j\ \forall i, j; \\ & s_i \neq s_j\ \forall i, j, i \neq j;\ (\cup_{i=1}^{r} R_i) \cup \left(\cup_{i=1}^{l} \{s_i\}\right) = S \Big\}.\end{aligned}$$

Ez olyan részleges szobabeosztások halmaza, ahol azon szobák száma, amelyekhez már hozzárendeltünk m hallgatót, r. Ekkor $\mathcal{P}_0$ egy egyelemű halmaz, amelynek eleme annak felel meg, amikor még egy hallgatót sem osztottunk be. $\mathcal{P}_k$ pedig a lehetséges (teljes) szobabeosztások halmaza, amelynek elemei azok a beosztások, ahol minden hallgatót hozzárendeltük valamelyik szobához.

7.3.2. Definíció. *Jelölje $k > 1$ a szobák számát, m a szobák (azonos) méretét és így $n = k \cdot m$ a hallgatók számát. Legyen $D \in \mathbb{R}^{n \times n}$ a hallgatók közötti euklideszi távolságok négyzeteinek mátrixa. Legyen $0 \leq r \leq k$ a már beosztott szobák száma, és ennek megfelelően legyen $P = (R_1, \dots, R_r, s_1, \dots, s_l) \in \mathcal{P}_r$ egy r-beosztás. Legyen $d : \mathcal{P}_r \to \mathbb{R}_0^+$ egy beosztás-költségfüggvény, amely megadja egy r-beosztás esetén a már beosztott szobákban lévő hallgatók szobán belüli távolságainak összegét:*

$$d(P) = \sum_{m=1}^{r} \sum_{\substack{i,j \in R_m, \\ i<j}} D_{ij}.$$

Legyen továbbá $\mathcal{D} = \{(P, d(P)) \,|\, P \in \mathcal{P}_r, 0 \leq r \leq k\}$ az összes lehetséges beosztás-költség pár halmaza, és jelölje $2^{\mathcal{D}}$ a $\mathcal{D}$ részhalmazainak halmazát. Végül pedig legyen $h : \mathcal{P}_r \times \mathbb{R}_0^+ \to 2^{\mathcal{D}}$ egy olyan szoba(halmaz)-hozzárendelés, ahol a $P \in \mathcal{P}_r$ r-beosztásra

$$\begin{aligned}h\left(P, d(P)\right) = \Big\{ & \left(P', d(P')\right) \,\Big|\, \left(P', d(P')\right) \in \mathcal{D}, \\ & P' = \left(R_1, \dots, R_r, R_{r+1}, s'_1, \dots, s'_{l-m}\right) \in \mathcal{P}_{r+1}, \\ & \min\{s_1, \dots, s_l\} \in R_{r+1} \Big\}.\end{aligned}$$

Az összes lehetséges szobabeosztás, azok költségeinek, valamint a minimum- és maxi-

mumértékek meghatározásához használt rekurzív eljárás pszeudokódját a 7.1. Algoritmus írja le.

Rekurzív eljárás a lehetséges szobabeosztások meghatározására

---

(1) *Változók inicializálása.*

$C = \emptyset$ [globális változó: az összes költség halmaza]

$Cmax = NaN, Cmin = NaN$ [globális változók: C szélsőértékei]

$r = 0$ [beosztott szobák száma]

(2) *A* **kroomcases**$(r, P, d(P))$ *rekurzív függvény futtatása.*

if $(r < k)$

$H \leftarrow h\big(P, d(P)\big)$ [$r+1$-szobabeosztások generálása]

for $\big(P', d(P')\big)$ in H

kroomcases$(r+1, P', d(P'))$

end for

else [költséghalmaz és szélsőértékek frissítése]

$C \leftarrow C \cup d(P)$

if (isnan$(Cmax)$) **or** $(Cmax < d(P))$

$Cmax \leftarrow d(P)$

end if

if (isnan$(Cmin)$) **or** $(Cmin > d(P))$

$Cmin \leftarrow d(P)$

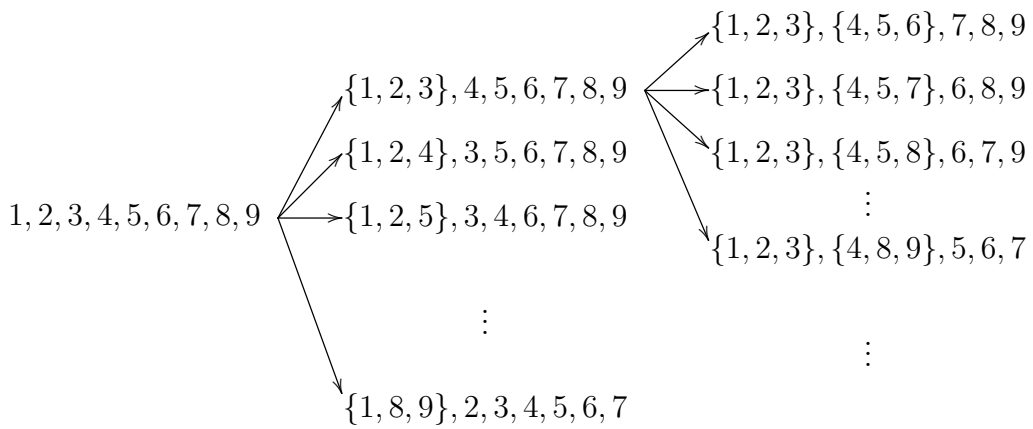
end if

end if

7.1. Algoritmus. Rekurzív eljárás a lehetséges szobabeosztások költségeinek és a költségek minimum és maximum értékeinek meghatározásához

Az algoritmus lényege - ahogyan az a definíciók alapján is sejthető -, hogy minden egyes lépésben kiválasztjuk a még nem csoportosított elemek közül a legkisebbet és e mellé a még nem csoportosított elemek közül az összes lehetséges módon választunk $m-1$ elemet visszatevés nélkül, ahol a sorrend nem számít. Ha az így kapott beosztásoknál még maradnak nem csoportosított elemek, akkor minden egyes így kapott részleges csoportosításra megismételjük a rekurzív lépést. Például $n = 9$ és $m = 3$ esetén a konstrukció a

7.7. ábra által bemutatott fa mentén halad.



7.7. ábra. A `kroomcases` eljárás rekurzív lépéseinek ábrázolása az összes lehetséges szobabeosztás megkonstruálására $n = 9$ és $m = 3$ esetén.

A konstrukció egyik előnye, hogy egy egyszerű és könnyen algoritmizálható metódust ad a szobabeosztások generálására. Másik pozitívuma pedig, hogy a szobabeosztások költségeinek számításánál minden lépésben elegendő csak az újonnan beosztott szobák költségeit meghatározni. Ez csökkenti a többszörös kalkulációk számát, ami jobb futásidőt eredményez.

7.3.3. Állítás. *A 7.1. Algoritmus az összes lehetséges szobabeosztást megkonstruálja, és mindegyiket pontosan egyszer.*

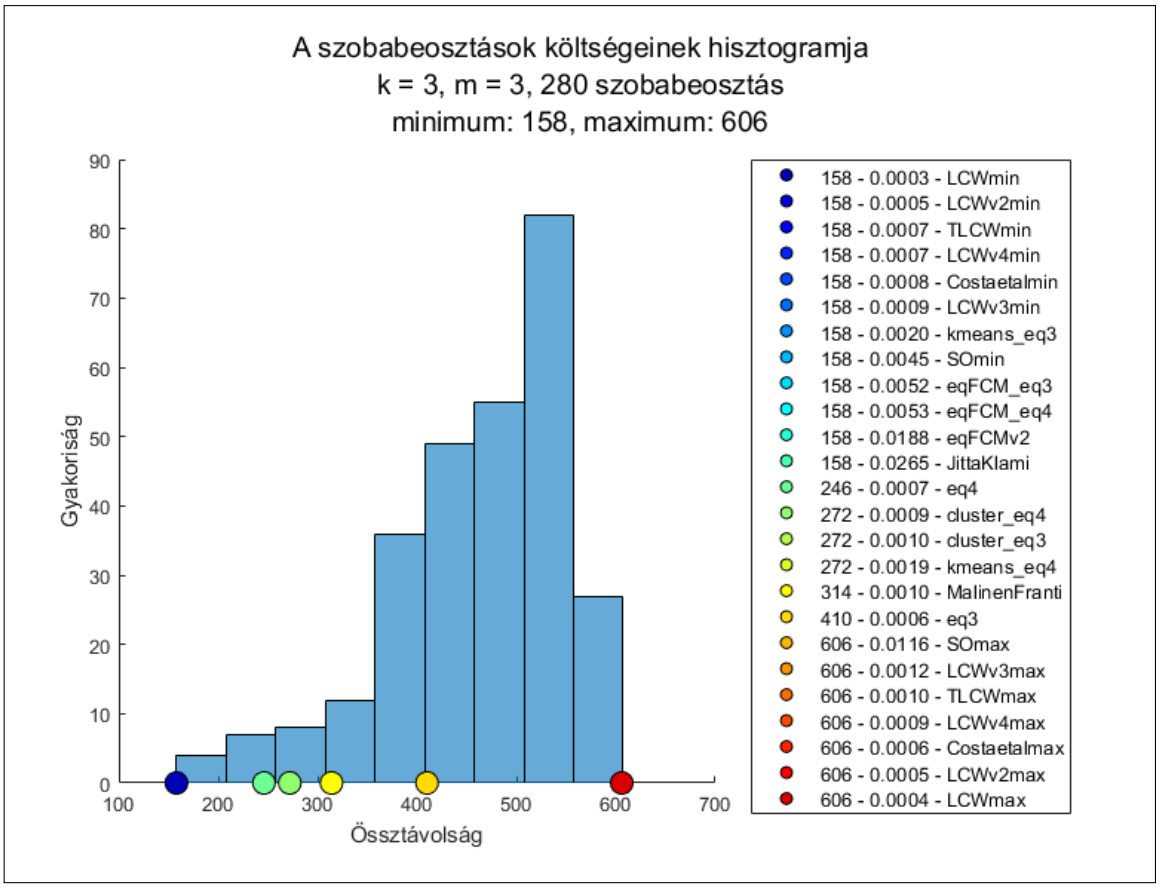
Bizonyítás. Könnyedén belátható, hogy a fán bármelyik kettő megkonstruált szobabeosztás különböző. Emellett a megkonstruált szobabeosztások száma

$$
\begin{aligned}
&1 \cdot \binom{n-1}{m-1} \cdot 1 \cdot \binom{n-m-1}{m-1} \cdot \ldots \cdot \binom{m-1}{m-1} = \\
&= \frac{(n-1)!}{(m-1)!(n-m)!} \cdot \frac{(n-m-1)!}{(m-1)!(n-2m)!} \cdot \ldots \cdot \frac{(2m-1)!}{(m-1)!m!} \cdot \frac{(m-1)!}{(m-1)!} = \\
&= \frac{(n-1)!}{((m-1)!)^{n/m}} \cdot \frac{1}{(n-m) \cdot (n-2m) \cdot (n-3m) \cdot \ldots \cdot m} = \\
&= \frac{n \cdot (n-1)!}{(m!)^{n/m} \cdot \frac{1}{m^{n/m}}} \cdot \frac{1}{n \cdot (n-m) \cdot \ldots \cdot m} = \frac{n!}{(m!)^{n/m}} \cdot \frac{1}{\frac{n}{m} \cdot \frac{n-m}{m} \cdot \ldots \cdot \frac{m}{m}} = \\
&= \frac{n!}{(m!)^{n/m} \cdot (n/m)!},
\end{aligned}
$$

hiszen minden rekurzív lépés során először egyértelműen a legkisebb, még nem csoportosított elemet választjuk, majd a többi még nem beosztott elem közül választunk $m - 1$

továbbit. A kapott kifejezés megegyezik a (3.1)-es fomula által megadott összes lehetőség számával, ebből pedig következik, hogy az algoritmus az összes különböző beosztást előállítja. □

7.3.2. Eredmények

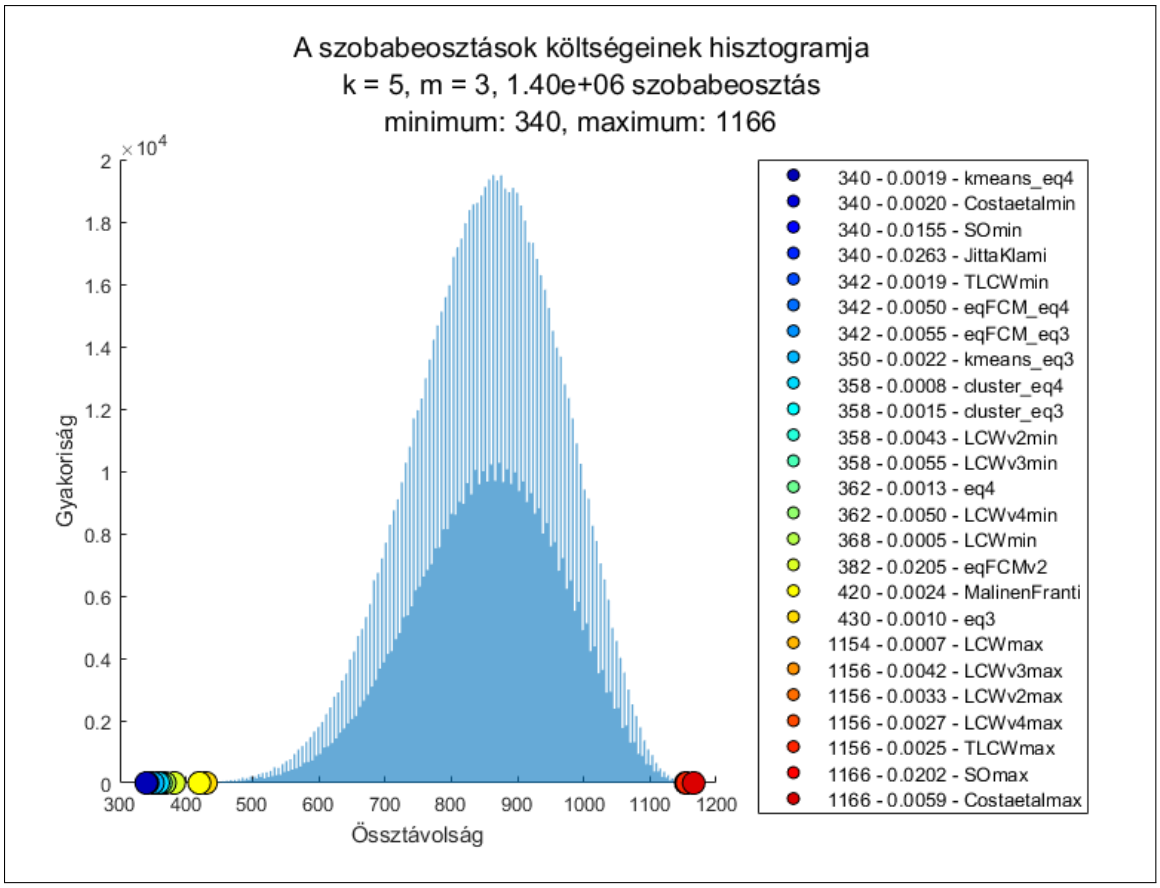


7.8. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 3, m = 3$, 280 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 0,1541 mp. Ferdeség: -0,5339. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

A 7.8. ábra $k = 3$ és $m = 3$ választás mellett egy szimulált adathalmaz esetében ábrázolja a lehetségek szobabeosztások költségeinek hisztogramját. Emellett megmutatja a beosztások költségének minimumát és maximumát, valamint azt is, hogy mik az egyes heurisztikák által megtalált célfüggvényértékek és futásidők. A feliratoknál először a minimumkereső algoritmusok vannak felsorolva célfüggvényérték és futásidő szerint is növekvő sorrendben, ezután pedig a maximumkereső algoritmusok következnek célfüggvényérték szerint szintén növekvő, de futásidő szerint csökkenő sorrendben. Ekképpen a legjobb

minimumkereső algoritmusok a lista elején, míg a legjobb maximumkereső algoritmusok a lista végén találhatóak.

Megfigyelhetjük, hogy 6 eljárás kivételével valamennyi, összesen 19 módszer megtalálja az optimumot, vagyis a megfelelő minimum vagy maximum értéket. A futásidők tekintetében észrevehetjük, hogy az egyes módszerek között nagyságrendbeli eltérések is megjelennek. Az `LCW` (minimumkereső és maximumkereső) algoritmusok kitűnnek azzal, hogy nagyon alacsony futásidő mellett is megtalálták az optimumot, de gyakorlatilag számos más eljárás is nagyon jól teljesít.



7.9. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 5, m = 3$, 1,40e+06 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 62,06 mp. Ferdeség: -0,2814. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

A hallgatók nagyobb számát tekintve, $k = 5$ és $m = 3$ esetét mutatja (1,40e+06 lehetséges szobabeosztásra) a 7.9. ábra.[4] Ekkor az optimumot megtaláló minimum- és

[4]További hasonló hisztogramok találhatóak a 7.3.3. Fejezetben, valamint az A.3. mellékletben. A fejezet 7.11. ábrája $k = 2$ és $m = 15$ paraméterekre (7,76e+07 lehetséges szobabeosztással) illusztálja az eredményeket egy szimulált hallgatói mintára. A mellékletben $k = 3$ és $m = 4$ értékek mellett

maximumkereső eljárások száma 10 (15 nem optimális eredmény ellenében). Emellett észrevehetjük még, hogy a teljesítményük szerint rendezett algoritmusok sorrendje is különböző.

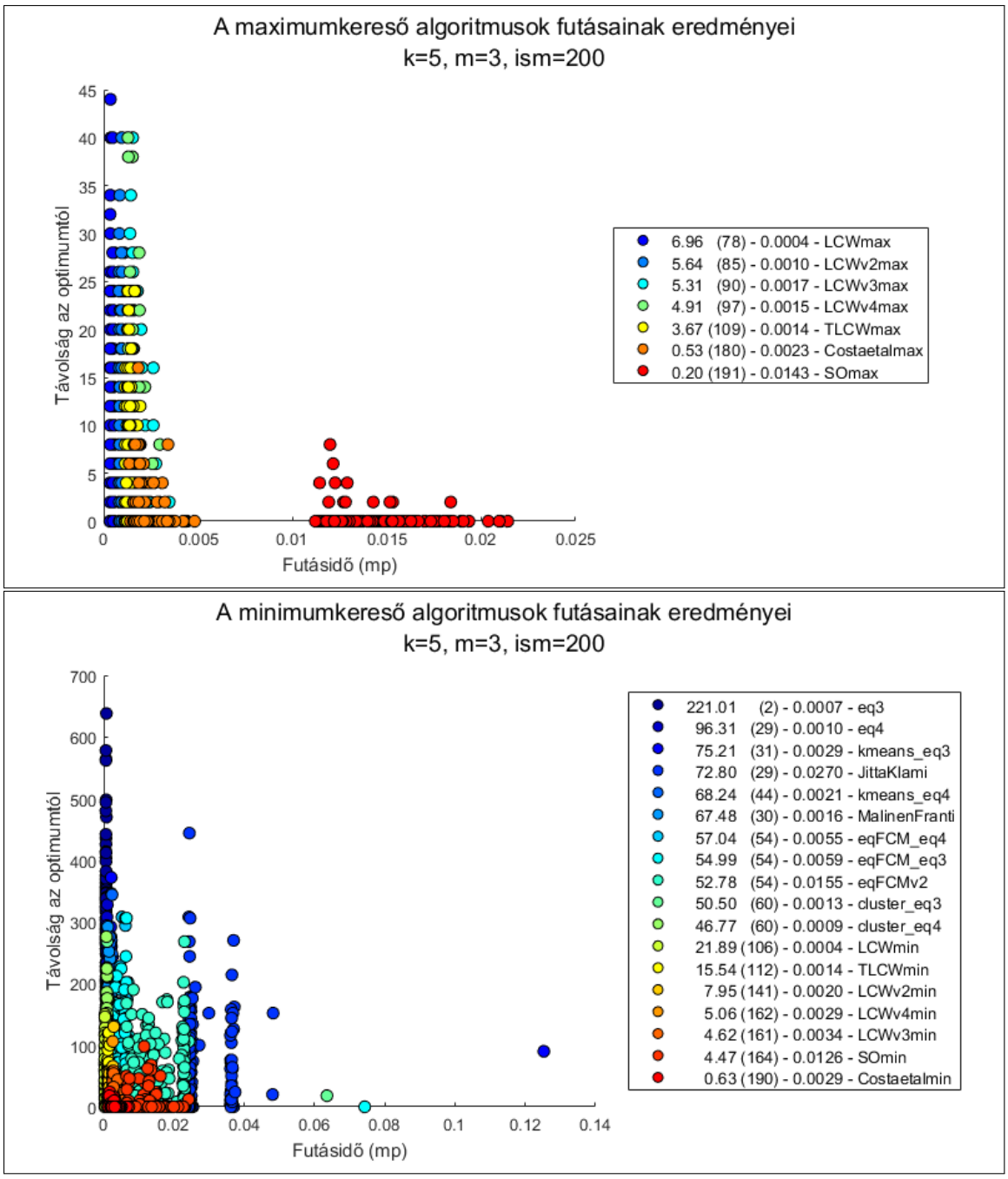
A robosztusabb elemzés érdekében különböző hallgatói mintákat is szimulálunk, és ezek mindegyikén kiértékeljük a heurisztikákat a tényleges optimum ellenében. A 7.10. ábra ennek a nagyobb, 200 elemű mintán (az ábrákon `'ism=200'`-ként jelölve) elvégzett elemzésnek az eredményeit mutatja be. Az alábrák feliratain megjelennek a számított statisztikák: az optimumtól való átlagos távolság (az első érték), hány esetben adta az algoritmus a legjobb eredményt (a második érték zárójelben) és az átlagos futásidő is (a harmadik érték). Az átlagokat a 7.2. Táblázat is tartalmazza a szórás értékekkel együtt, a minimumkereső eljárások esetében ugyanakkor csak a legjobban teljesítő módszerek értékei vannak feltüntetve.

A 7.10. ábránál a maximumkereső eljárások között egyértelműen azok teljesítenek jobban, amelyek szofisztikáltabb javító módszert alkalmaznak a megoldás keresése során. Ezek az `SOmax` és `Costaetalmax` algoritmusok. Érdemes ugyanakkor azt is megfigyelni, hogy az `SOmax` eljárás futásideje egy nagyságrenddel nagyobb, mint az azt követő `Costaetalmax` módszeré. A hármas cseréket megvalósító algoritmusok tekintetében az `LCWv4max` teljesít a legjobban.

A minimumkereső eljárások ábrájánál szintén a szofisztikált módszerek érték el a legjobb eredményeket. Emellett azt vehetjük észre, hogy a konstruktív módszerek, a klaszterelemzéshez kapcsolódó heurisztikák, a `JittaKlami`, valamint a `MalinenFranti` algoritmusok a legjobban teljesítő módszerektől jóval lemaradva, legfeljebb az esetek 30%-ában találták meg a legjobb megoldást. Továbbá a hármas cseréket megvalósító eljárások között az átlagos távolságok szempontjából az `LCWv3min` teljesít a legjobban, míg ha azt nézzük, hogy melyik módszer hányszor találta meg a legjobb megoldást, akkor bár csak egy hajszálnyival, de az `LCWv4min` módszer nyer. Futásidő szempontjából viszont a kettő közül egyértelműen az utóbbi a győztes.

Hogy sokkal jobb rálátásunk legyen a jól teljesítő algoritmusok működésére, az elemzést

(5775 lehetséges szobabeosztással) a A.5. ábra, $k = 4$ és $m = 3$ paraméterek esetén (12 fő és 15400 lehetséges szobabeosztás) a A.6. ábra, míg $k = 6, m = 3$ értékek esetén (18 hallgató, 1,91e+08 lehetséges szobabeosztás) a A.7. ábra mutatja be az eredményeket egy-egy szimulált hallgatói mintára.



7.10. ábra. A maximum- és minimumkereső algoritmusok futásainak eredményei. $k = 5, m = 3$, 200 szimulált eset. A feliratok magyarázata: a célfüggvényérték átlagos távolsága az optimumtól (azon esetek száma, amikor a célfüggvényérték megegyezik az optimummal) - átlagos futásidő (mp) - algoritmus neve.

	Távolságok	Futásidők	# legjobb eredmény
LCWmax	6,96	**0,0004**	78
	(9,03)	***(0,0001)***	
LCWv2max	5,64	0,0010	85
	(7,71)	*(0,0002)*	
LCWv3max	5,31	0,0017	90
	(7,65)	*(0,0004)*	
LCWv4max	4,91	0,0015	97
	(7,52)	*(0,0003)*	
TLCWmax	3,76	0,0014	109
	(5,73)	*(0,0002)*	
Costaetalmax	0,53	0,0023	180
	(1,85)	*(0,0006)*	
SOmax	**0,20**	0,0143	**191**
	(0,92)	*(0,0022)*	

	Távolságok	Futásidők	# legjobb eredmény
LCWmin	21,89	**0,0004**	106
	(32,92)	***(0,0001)***	
TLCWmin	15,54	0,0014	112
	(25,38)	*(0,0003)*	
LCWv2min	7,95	0,0020	141
	(16,88)	*(0,0006)*	
LCWv3min	4,62	0,0034	161
	(11,85)	*(0,0009)*	
LCWv4min	5,06	0,0029	162
	(13,98)	*(0,0005)*	
SOmin	4,47	0,0126	168
	(13,60)	*(0,0035)*	
Costaetalmin	**0,63**	0,0029	**190**
	(3,02)	*(0,0009)*	

7.2. Táblázat. A maximum- (felül) és minimumkereső (alul) algoritmusok futásainak eredményei: átlagos távolság az optimális megoldástól (és szórása), átlagos futásidő (és szórása), valamint azon esetek száma, amikor az adott algoritmus adta a legjobb eredményt. $k = 5, m = 3$, 200 szimulált eset.

az elemszámok növelése mellett a 7.4. fejezetben folytatjuk. Ezt az is indokolja, hogy a megfogalmazott m-szobatárs probléma megoldhatóságát olyan modellben is vizsgáljuk, amely tükrözi egy valódi kollégium férőhelyeinek számát. Elvégre a 15 fős kollégiumok nem túl gyakoriak.

Az előzőekben tárgyaltak következtében a további elemzésből kihagyjuk a hagyományos klaszterelemzéshez kapcsolódó eljárásokat, név szerint az `eq3`, az `eq4` módszereket, valamint a `cluster`, a `kmeans` és az `eqFCM` klaszterező eljárásokat valamennyi kiegyenlítő módszer esetén. Emellett nem szerepeltetjük tovább az `eqFCMv2`, a `JittaKlami` és a `MalinenFranti` algoritmusokat sem. Ezen módszerek gyengébb teljesítménye azt mutatja, hogy a viszgált optimalizálási probláma megoldására általánosan nem teljesítenek jól azok az eljárások, amelyek a klaszterközéppontok meghatározásának és az elemek hozzárendelésének iteratív váltakozásain alapszanak.

Így a bemutatott eredmények tükrében tulajdonképpen a lokális kereső módszerek további vizsgálatával folytatjuk. Ugyanakkor a továbbiakban a hármas cserék közül csak egyet vizsgálunk, amely az `LCWv4` módszer. Ennek okai a következőek. Egyrészt szeretnénk a (legfeljebb) páros cseréket megvalósító heurisztikák által adott megoldásokat kezdőmegoldásként használni egy hármas cseréket is alkalmazó módszernél, hogy megvizsgáljuk a javítás mértékét. A túl sok kombináció elkerülése érdekében ezt csak egy `LCW` változattal tesszük meg. Másrészt az elemszámok növekedése miatt a futásidők drasztikus emelkedése várható. Az összfutásidő csökkentése érdekében szintén indokolt az elemzésben lévő eljárások számának csökkentése. A választás azért az `LCWv4` módszerre esett, mert ez összességében kicsivel jobban teljesített, mint az `LCWv2` és `LCWv3` heurisztikák.

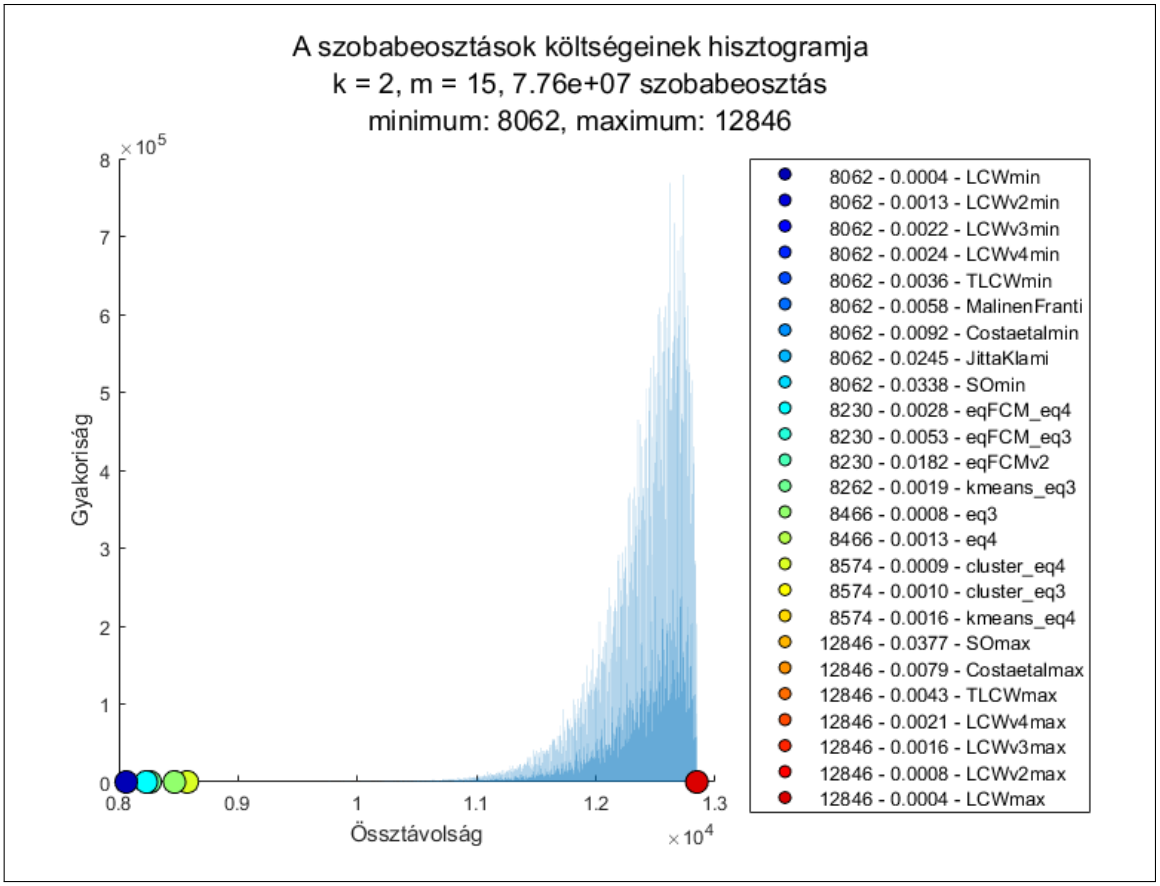
Végül megjegyezzük, hogy az egyszerű `LCW` módszert megtartjuk a továbbiakban is, mivel nagyon kedvező futásidők mellett elfogadható eredményeket produkált.

7.3.3. A minimalizálási és maximalizálási feladatok aszimmetriája

A fejezetet egy érdekes megfigyelés leírásával zárjuk, amelynek egyértelmű szemléltetéséhez a 7.11. ábrát használjuk fel. Az ábra $k = 2, m = 15$ paraméterek esetén mutatja a lehetséges szobabeosztások költségeinek hisztogramját. Vegyük észre, hogy az eloszlás bal széle rendkívül vékony a jobb széléhez képest. Ebben az esetben azt mondjuk, hogy az eloszlás balra ferde, amelyet a kiszámított ferdeség érték negatív mivolta is megerősít: -0,9044. Ez a kereső eljárások szempontjából a minimalizálási és maximalizálási problémák

lehetséges aszimmetriájára hívja fel a figyelmet.

Az eddig bemutatott, hisztogrammal ábrázolt valamennyi minta ferdesége negatív. Emellett több paraméterpárra ($k = 3, m = 3$; $k = 3, m = 4$; $k = 4, m = 3$; $k = 5, m = 3$) is egyenként 200 hallgatói mintát generálva, és az egyes minták esetén a szobabeosztások költségeit meghatározva a ferdeségi értékek kivétel nélkül mind negatívak voltak.



7.11. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 2, m = 15$, 7,76e+07 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 54 p 6 mp. Ferdeség: $-0{,}9044$. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

7.4. Nagyobb hallgatói elemszámok

A lokális kereséseket megvalósító eljárásokat további vizsgálatoknak vetjük alá, amelyet a korábbitól eltérő módszerrel teszünk meg a fejezet során. Ennek oka elsősorban, hogy a nagyobb hallgatói minták esetén az optimumok keresésének futásideje az összes lehetséges szobabeosztás költségét végignézve túl nagyra nő.

A $k = 2$ és $m = 15$ paraméterek mellett (30 hallgató és 7,76e+07 lehetséges szobabe-

osztás esetén, lásd a 7.11. ábrát) a minimum és maximum meghatározása csaknem 1 órát vesz igénybe. Ha a futásidőt állandónak tekintjük, akkor 200 hallgatói minta generálása és kiértékelése a tényleges optimumok ellenében több, mint egy hétig tartana.[5]

Másodlagos szempont pedig, hogy a szobabeosztások költségeinek eloszlását szemléltető hisztogramot nem tudunk előállítani, mert a számítógép - amelyen a kísérletek futottak - memóriája nem képes tárolni egyben a szobabeosztások költségeinek vektorát.[6, 7]

A következőekben $n = 600$ fős hallgatói mintákat generálunk, amelyekre $m = 2$, $m = 5$, valamint $m = 60$ fős csoportokat feltételezve teszteljük a kiválasztott algoritmusokat. Az elsődleges célunk annak vizsgálata, hogy a hármas cseréket implementáló `LCWv4` eljárás mennyiben járul hozzá a meglévő algoritmusok köréhez az optimális megoldás keresésében. Ehhez amellett, hogy az algoritmust önmagában is futtatjuk, a `TLCW`, az `SO` és a `Costaetal` algoritmusok által adott megengedett megoldásokat kezdőértékként használva, az `LCWv4` módszer segítségével megkíséreljük tovább javítani ezen heurisztikák eredményeit. Ezt az ábrák feliratain a nevek kombinálásával jelöljük, például `SOminLCWv4` mutatja hogy az `SOmin` heurisztika eredményére alkalmaztuk az `LCWv4min` eljárást.

Párokra, vagyis $m = 2$ esetén a heurisztikák eredményeit összevetjük az optimális megoldással, amelyet egy Edmonds algoritmusára épülő eljárással határozunk meg (az implementációért lásd Saunders (2022)). Az eljárásokat egymáshoz képest értékeljük ki a 7.4.1. fejezetben bevezetésre kerülő *optimalitási pontszám* felhasználásával, amely különösen hasznos lesz az általános esetben (vagyis $m > 2$ esetén), amikor a tényleges optimum meghatározása nem megvalósítható

7.4.1. Optimalitási pontszám

7.4.1. Definíció. *Jelölje M a hallgatói minták számát. Jelölje továbbá x_i^{max} és x_i^{min} a heurisztikák által megtalált legnagyobb, illetve legkisebb célfüggvényértékeket az i-edik minta esetén. Tegyük fel, hogy $x_i^{max} > x_i^{min}$ $\forall i$. Legyen végül x_i^{alg} egy tetszőleges 'alg' algorit-*

[5] Az A.3. mellékletben megtalálható egy, a $k = 6, m = 3$ értékek esetén (18 hallgató, 1,91e+08 lehetséges szobabeosztás) szimulált hallgatói minta hisztogramja, amelyre az optimumok keresése ezt az időt is meghaladta, és több, mint 2 órát vett igénybe.

[6] $k = 5, m = 4$ paraméterek esetén (20 fő, 2,5e+09 lehetséges szobabeosztás) 19,0 GB memóriára lenne szükség, míg $k = 7, m = 3$ értékekre (21 fő, 3,6e+10 lehetséges szobabeosztás) már 269,8 GB szükséges.

[7] Ha szemléltetni szeretnénk az eloszlást, akkor választhatjuk megoldásképpen, hogy véletlenszerűen generálunk szobabeosztásokat és kiértékeljük azok költségét. Ha ezt kellőképpen nagy számban tesszük meg, és ábrázoljuk az így kapott értékek hisztogramját, akkor az várhatóan tükrözi majd a szobabeosztások költségeinek tényleges eloszlását. Persze a hasonlóság mértéke nagyban függ attól, hogy mekkora a különbség a véletlen minta elemszáma, és a lehetséges beosztások tényleges száma között.

mus által megtalált célfüggvényérték az i-edik minta esetén. Ekkor az 'alg' algoritmus p^{alg} optimalitási pontszáma

$$p^{alg} = \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{alg} - x_i^{\min}}{x_i^{\max} - x_i^{\min}}.$$

7.4.2. Állítás. *Az optimalitási pontszámra az alábbi tulajdonságok teljesülnek:*

1. *Bármely algoritmus esetén $p^{alg} \in [0, 1]$.*
2. *Két különböző 'alg1' és 'alg2' algoritmusra $p^{alg1} \leq p^{alg2}$ akkor és csak akkor, ha*

$$0 \leq \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{alg2} - x_i^{alg1}}{x_i^{\max} - x_i^{\min}}.$$

3. *Tetszőleges maximumkereső 'alg' algoritmusra*

$$\frac{1}{M} \sum_{i=1}^{M} \left(x_i^{\max} - x_i^{alg} \right) \leq \max_i \left\{ x_i^{\max} - x_i^{\min} \right\} \left(1 - p^{alg} \right).$$

Hasonlóan, tetszőleges minimumkereső 'alg' algoritmusra

$$\frac{1}{M} \sum_{i=1}^{M} \left(x_i^{alg} - x_i^{\min} \right) \leq \max_i \left\{ x_i^{\max} - x_i^{\min} \right\} p^{alg}.$$

Bizonyítás.

1. és 2. pont triviális.

3. Egy '*alg*' maximumkereső algoritmusra a következő teljesül:

$$\begin{aligned}
\frac{1}{M} \sum_{i=1}^{M} \left(x_i^{\max} - x_i^{alg} \right) &= \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{\max} - x_i^{alg}}{x_i^{\max} - x_i^{\min}} \left(x_i^{\max} - x_i^{\min} \right) \leq \\
&\leq \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{\max} - x_i^{alg}}{x_i^{\max} - x_i^{\min}} \max_i \left\{ x_i^{\max} - x_i^{\min} \right\} = \\
&= \max_i \left\{ x_i^{\max} - x_i^{\min} \right\} \left(1 - \frac{1}{M} \sum_{i=1}^{M} \frac{x_i^{alg} - x_i^{\min}}{x_i^{\max} - x_i^{\min}} \right) = \\
&= \max_i \left\{ x_i^{\max} - x_i^{\min} \right\} \left(1 - p^{alg} \right).
\end{aligned}$$

Hasonlóan következik az egyenlőtlenség minimumkereső algoritmus esetében is. □

Az optimalitási pontszám lényege, hogy különböző csoportszámok (k) és szobaméretek (m) mellett is azonos skálán tudjuk összehasonlítani az eredményeket. A hallgatói mintánkénti max-min különbséggel, vagyis az $\left(x_i^{\max} - x_i^{\min} \right)$ értékkel való skálázás[8] pedig azért indokolt, mert már azonos k és m értékek esetén is eltérő lehet a szobabeosztások költségeihez tartozó eloszlás szélessége. A 7.4.2. Állítás 1. pontja jelöli a közös skálát.

[8]Megjegyezzük, hogy a kísérletek során $x_i^{\max} - x_i^{\min} > 0 \ \forall i$ minden k és m érték esetén teljesült.

A 2. pontban megfogalmazott tulajdonság alapján látható, hogy amikor az optimalitási pontszám alapján fogalmazunk meg sorrendet, akkor gyakorlatilag az algoritmusok eredményeinek átlagos skálázott különbsége alapján tesszük meg ezt. Ennek megfelelően azt mondjuk, hogy az a maximumkereső (minimumkereső) eljárás teljesít a legjobban, amelynek pontszáma a legközelebb áll 1-hez (0-hoz).

Végül a 3. pont az optimalitási mutató interpretálásához nyújt segítséget. Ugyanis egy tetszőleges maximumkereső algoritmus és a megtalált legnagyobb célfüggvényérték közötti átlagos különbség értékét felülről becsülhetjük az optimalitási pontszám segítségével. Így egy egyszerű formában lehet egy képünk az algoritmusok és az elérhető legjobb eredmény viszonyáról. Továbbá ezzel összhangban hivatkozunk arra, hogy egy 'alg' maximumkereső algoritmus $(1 - p^{alg}) \cdot 100\%$-ra van a legjobban teljesítő eljárástól, és minimumkereső algoritmusra hasonlóan.

7.4.2. Edmonds algoritmusa mint viszonyítási alap

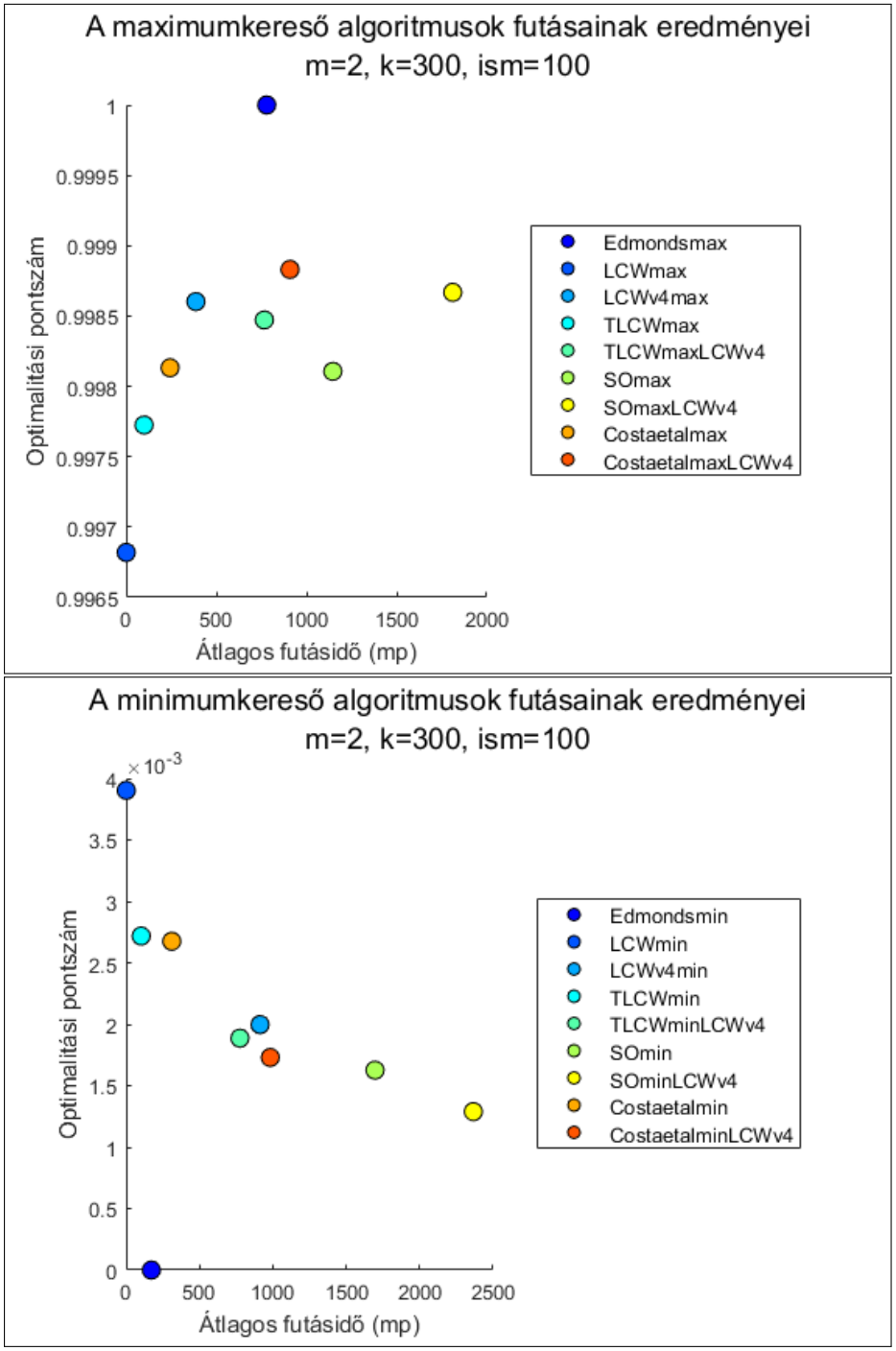
Párok esetében ismert, hogy polinomiális időben meg tudjuk határozni az optimális megoldást. Emiatt ebben az esetben a heurisztikus eljárások eredményét össze tudjuk hasonlítani a tényleges optimummal. Ebből persze nem tudunk megbízható következtetéseket levonni arra vonatkozóan, hogy a heurisztikus eljárások hogyan teljesítenének általános esetben, ugyanakkor bizakodásra adhat okot, ha azt látjuk, hogy relatíve közel tudunk kerülni az optimumhoz.

A 7.12. ábra 100 generált hallgatói minta esetén mutatja az eredményeket $m = 2$ hallgatóból álló csoportok mellett, ahol a csoportok száma $k = 300$. Az eljárások között szerepeltetünk egy-egy Edmonds algoritmusára építő minimum-, illetve maximumkereső eljárást, amelyekre rendre `Edmondsmin` és `Edmondsmax` néven hivatkozunk[9]

A 7.12. ábra alapján azt mondhatjuk, hogy relatíve közel tudunk kerülni az optimumhoz, akár még önmagában az `LCW` eljárással is. Emellett a szofisztikáltabb módszerek jelentősen tudnak javítani a célfüggvényértéken, ugyanakkor úgy tűnik, hogy még ezekkel sem tudjuk teljesen lefaragni az `LCW` megoldása és az optimum közötti különbséget. A heurisztikák részletes elemezése jelen esetben nem célunk, ugyanakkor megjegyezzük, hogy a maximumkereső eljárásoknál az `LCWv4` módszer önmagában is relevánsnak tűnik, a minimumkereső eljárásoknál ugyanakkor valamivel rosszabbul teljesített, mint a

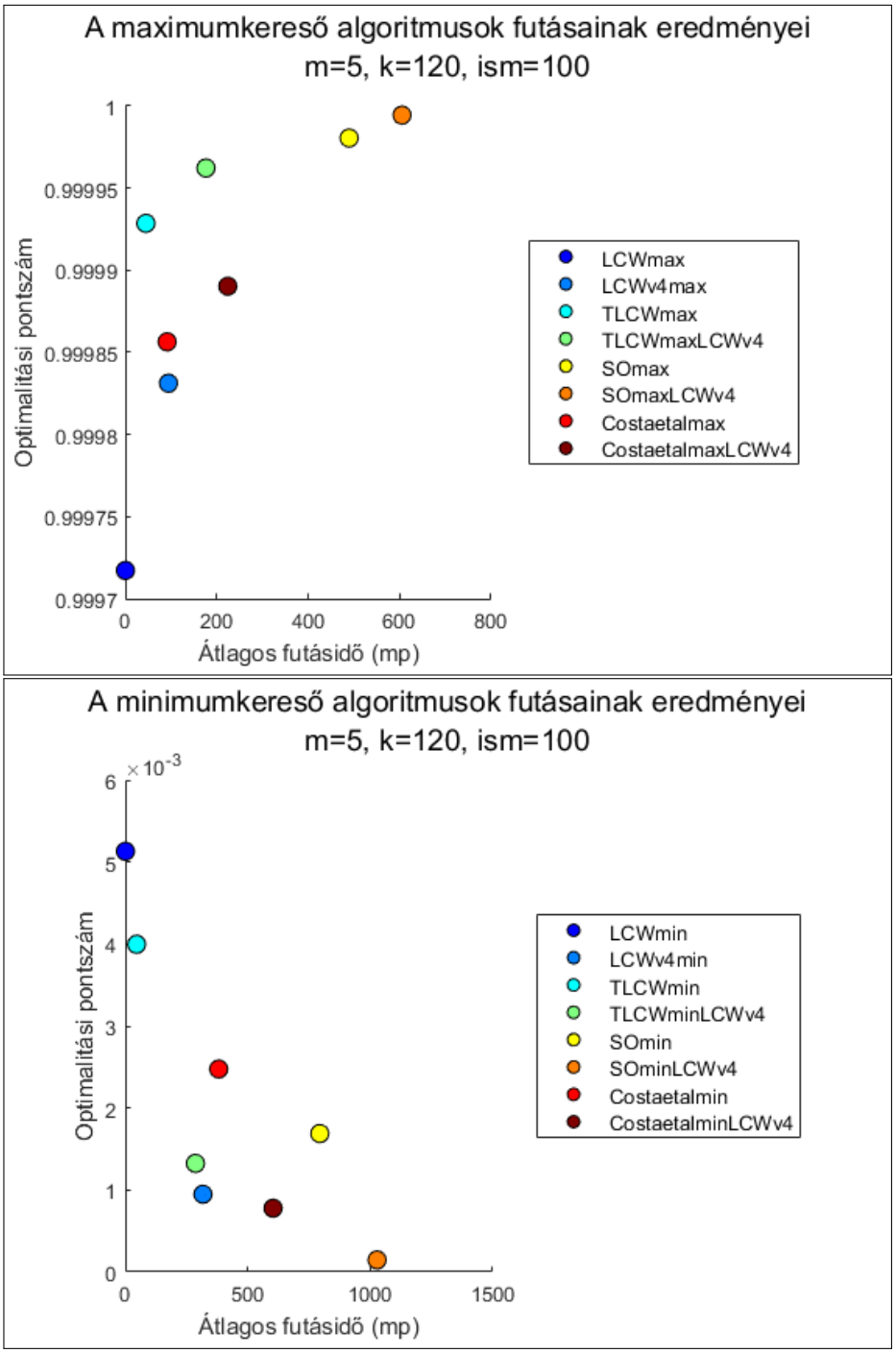
[9] Az algoritmusok MATLAB kódja nyíltan hozzáférhető, lásd Saunders (2022).

`TLCWminLCWv4` és `CostaetalminLCWv4` módszerek. Valamely másik heurisztika után alkalmazva az `LCWv4` a legtöbb esetben jelentősen tud javítani a célfüggvényértéken.



7.12. ábra. A maximum- és minimumkereső algoritmusok futásainak eredményei, Edmonds algoritmusát is beleértve. $m = 2, k = 300$, 100 szimulált eset.

7.4.3. Eredmények legalább háromfős csoportokra



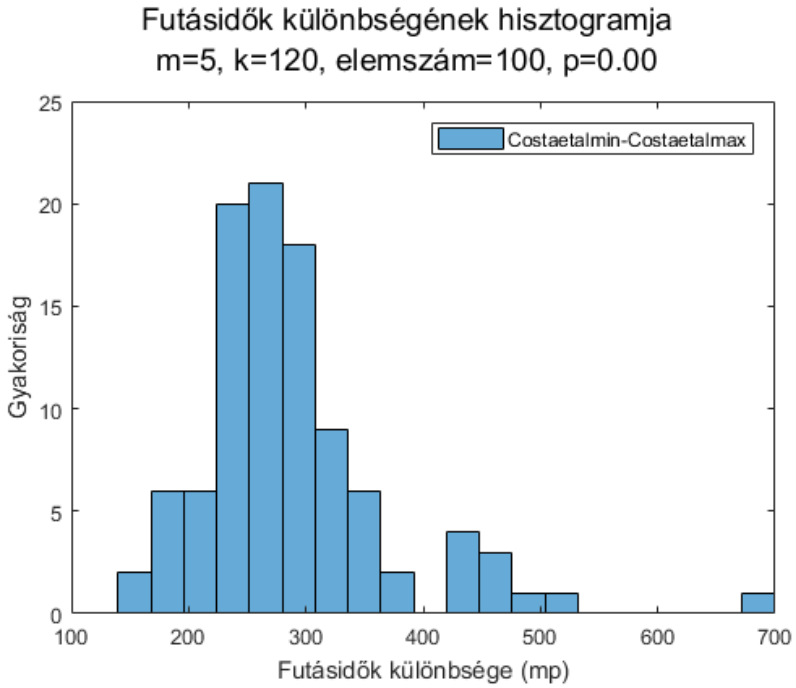
7.13. ábra. A maximum- és minimumkereső algoritmusok futásainak eredményei. $m = 5, k = 120$, 100 szimulált eset.

A 7.13. ábra 100 generált hallgatói minta esetén mutatja az eredményeket $m = 5$ hallgatóból álló csoportok mellett, ahol a csoportok száma $k = 120$. A maximumkereső eljárások esetében elsőként azt figyelhetjük meg, hogy az optimalitási pontszám tekintetében nagyon kicsik a különbségek az egyes eljárások átlagos eredményei között. Még a

legrosszabbul teljesítő `LCWmax` módszer is kevesebb, mint 0,03%-ra van a legjobban teljesítő eljárástól.

Emellett észrevehetjük, hogy az `LCWv4max` módszer önmagában redundáns, hiszen a `TLCWmax` módszer mind futásidő, mind optimalitási pontszám tekintetében jobban teljesít. Ugyanakkor a heurisztika alkalmazása más szofisztikált keresőmódszerek eredményeire tovább javította azok célfüggvényértékét. Továbbá megjegyezzük, hogy az eredmények alapján a maximalizálási problémára a legjobb eredményeket a `TLCWmax` és az `SOmax` algoritmusok adják.

A minimumkereső módszerek eredményei felé fordulva az optimalitási pontszám szempontjából az előzőekhez hasonlóan, egymáshoz közeli eredményeket látunk. Ebben az esetben az `LCWv4min` eljárásnak ugyanakkor már önmagában is van relevanciája, hiszen futásidő és optimalitási pontszám szerint is jobban teljesít, mint a `Costaetalmin` vagy az `SOmin` módszer. Emellett más heurisztikák eredményeit kezdőértékként felhasználva jelentősen tudott javítani azok célfüggvényértékén.



7.14. ábra. A futásidők különbségének hisztogramja a `Costaetal` módszer minimum- és maximumkereső változatai esetén. $m = 5, k = 120$, 100 szimulált eset. H0: A különbségek átlaga nulla.

Végül vegyük észre, hogy a minimalizálási probléma esetén az algoritmusok futtatása jelentősen több időt igényel, mint a maximalizálási probléma során. Például vegyük a `Costaetal` algoritmus minimalizálási és maximalizálási verziója esetén a futásidők különb-

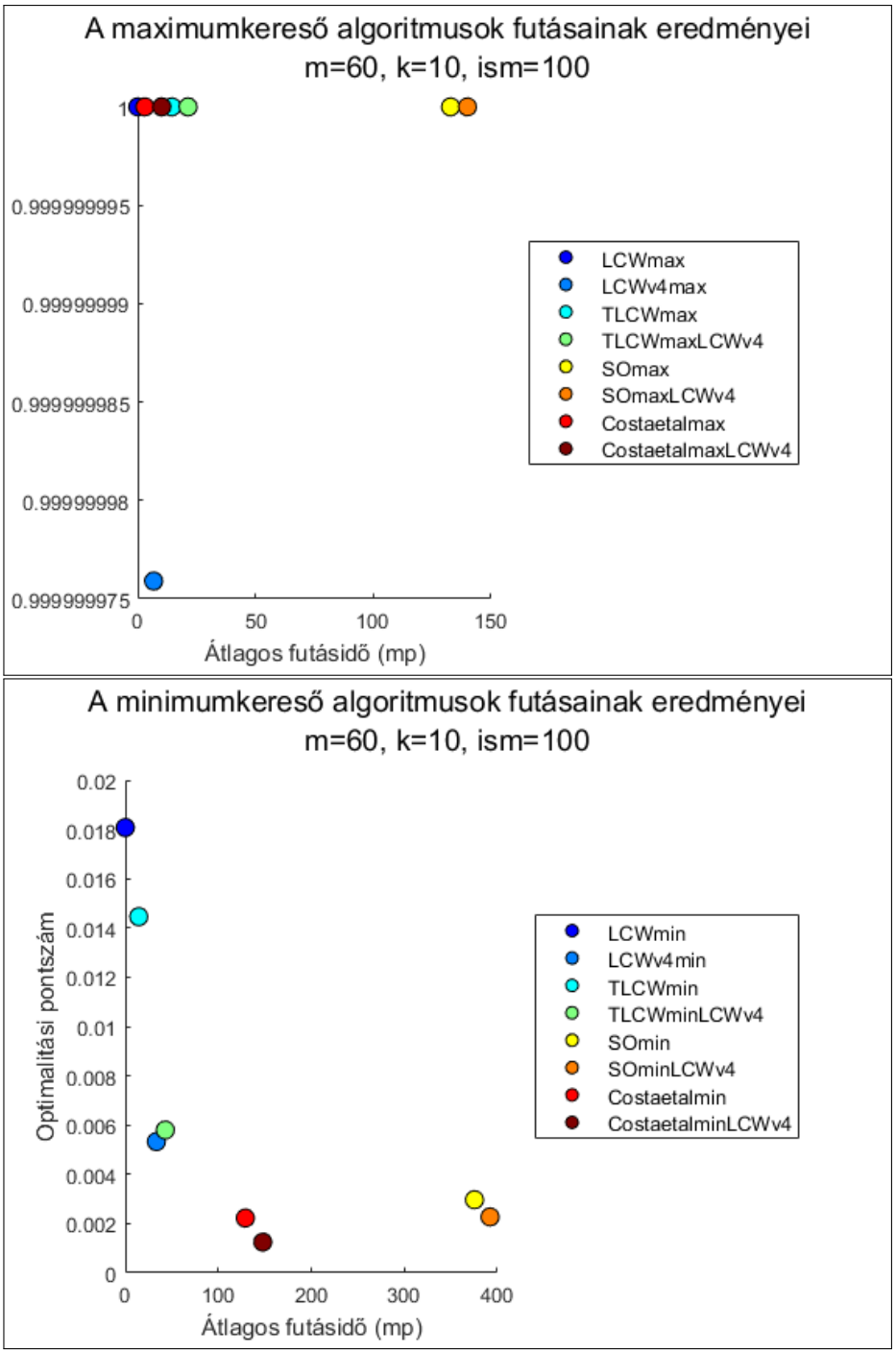
ségeit, amelyet hisztogram formájában a 7.14. ábra mutat be. Látható, hogy valamennyi generált minta esetén pozitív a különbség a minimalizálási probléma javára. Ennélfoga egymintás t-próbával tesztelve a H0 hipotézist, miszerint a különbségek átlaga 0, természetesen bármilyen szignifikancia-szint mellett elvethetjük. A jelenség fennáll a többi eljárás esetében is, így ez is tükrözi a minimalizálási és maximalizálási feladatok aszimmetriáját.

Az eljárások eredményeit a szórásértékekkel együtt (zárójelben) a 7.3. Táblázatban is közöljük. Az algoritmusok optimalitási pontszám és futásidő szerint vannak rendezve úgy, hogy fentről lefelé haladva láthatóak a jobban teljesítő módszerek. A redundáns algoritmusokat szürke háttérrel jelöltük. A táblázat alapján jól látható, hogy ezekre létezik legalább olyan jó optimalitási pontszámot, ugyanakkor kevesebb idő alatt elérő, másik heurisztika.

	Optimalitási pontszám	Futásidő
LCWmax	0,99972	0,58
	(0,00011)	*(0,02)*
LCWv4max	0,99983	94,60
	(0,00009)	*(20,08)*
Costaetalmax	0,99986	91,72
	(0,00007)	*(25,87)*
CostaetalmaxLCWv4	0,99989	224,37
	(0,00006)	*(46,75)*
TLCWmax	0,99993	45,22
	(0,00006)	*(0,60)*
TLCWmaxLCWv4	0,99996	176,57
	(0,00004)	*(38,48)*
SOmax	0,99998	489,61
	(0,00002)	*(79,18)*
SOmaxLCWv4	0,99999	605,27
	(0,00001)	*(90,75)*

	Optimalitási pontszám	Futásidő
LCWmin	0,00513	1,05
	(0,00091)	*(0,13)*
TLCWmin	0,00400	46,54
	(0,00092)	*(0,66)*
Costaetalmin	0,00247	381,77
	(0,00074)	*(79,86)*
SOmin	0,00169	794,44
	(0,00049)	*(137,72)*
TLCWminLCWv4	0,00132	286,94
	(0,00075)	*(45,30)*
LCWv4min	0,00095	317,58
	(0,00063)	*(43,61)*
CostaetalminLCWv4	0,00078	603,83
	(0,00067)	*(86,85)*
SOminLCWv4	0,00015	1028,23
	(0,00026)	*(137,40)*

7.3. Táblázat. Maximum- (fent) és minimumkereső (lent) algoritmusok eredményei, $m = 5, k = 120$. Szürkével jelölve a redundáns heurisztikákat.

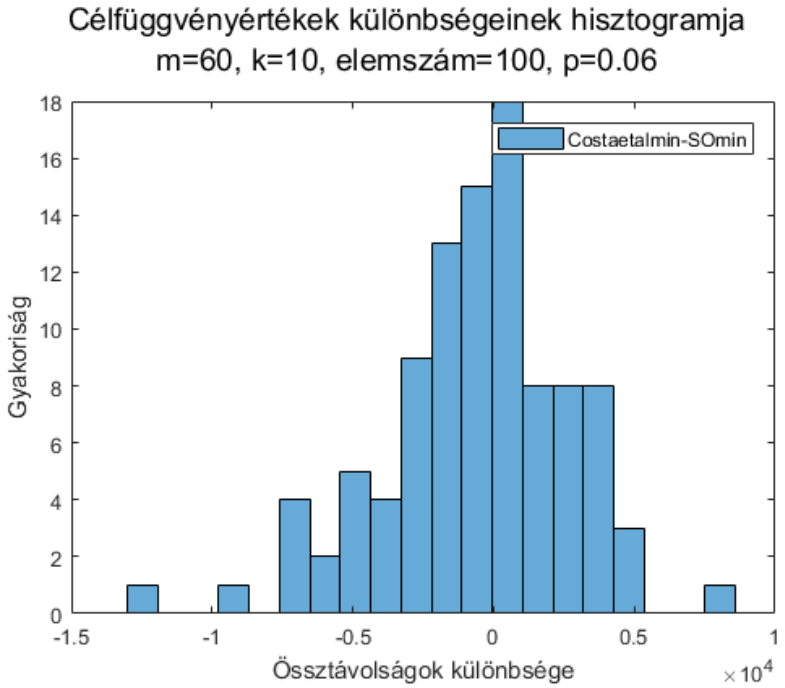


7.15. ábra. A maximum- és minimumkereső algoritmusok futásainak eredményei. $m = 60, k = 10$, 100 szimulált eset.

A 7.15. ábra $m = 60$ főből álló csoportok és $k = 10$ csoport mellett mutatja az eredményeket 100 generált hallgatói mintából számítva. A maximumkereső eljárások esetében gyakorlatilag minden eljárás minden esetben megtalálta ugyanazt az optimumot - az `LCWv4` eredmény 1e-8 nagyságrendű eltérésének hátterében valószínűleg numerikus pontatlanság áll. Ennélfogva hasonló esetekben még az egyszerű `LCWmax` eljárás is ele-

gendő lehet az optimum keresésére. Vegyük észre, hogy az állítás nem az, hogy biztosan megtalálja az optimumot, hanem az, hogy várhatóan hasonlóan jól teljesít, mint a szofisztikáltabb algoritmusok, ráadásul alacsonyabb futásidő mellett.

A minimumkereső heurisztikák ábrája esetében azt láthatjuk, hogy az optimalitási pontszám mentén az értékek sokkal jobban szétterülnek, mint a csoportok nagy száma esetén. A legjobban és legrosszabbul teljesítő módszerek optimalitási pontszáma közötti különbség körülbelül 1,8%. Ez azt mutatja, hogy ezen algoritmusok köréből a legegyszerűbb módszert választva is relatíve közel vagyunk a legszofisztikáltabb eljárások teljesétményéhez.



7.16. ábra. A célfüggvényértékek különbségének hisztogramja a `Costaetalmin` és az `SOmin` módszerek esetén. $m = 60, k = 10$, 100 szimulált eset. H0: A különbségek átlaga nulla.

Az eredmények alapján az `SOmin` és `Costaetalmin` módszerek hasonlóan teljesítenek. A 7.16. ábra mutatja a célfüggvényértékek különbségének hisztogramját, és az egymintás t-statisztika értékét arra a nullhipotézisre, miszerint a különbségek átlaga nulla. Ez alapján 95%-os szignifikancia-szint mellett még nem vethető el a nullhipotézis (p=0,06), ugyanakkor a futásidőkbeli jelentős különbség miatt a `Costaetalmin` eljárás egyértelműen jobbnak tűnik. Emellett megfigyelhetjük, hogy az `LCWv4min` módszer segítségével még ennek célfüggvényértékén is tudtunk javítani. Megjegyezzük továbbá, hogy habár az `LCWv4min` jobbnak tűnik a `TLCWminLCWv4` módszernél, azok eredményei elég közel esnek egymáshoz. Az eljárások eredményeit a 7.4. Táblázatban is közöljük.

	Optimalitási pontszám	Futásidő
SOmaxLCWv4	1,00	139,96
	(0,00)	*(0,55)*
SOmax	1,00	132,79
	(0,00)	*(0,55)*
TLCWmaxLCWv4	1,00	21,39
	(0,00)	*(0,11)*
TLCWmax	1,00	14,52
	(0,00)	*(0,09)*
CostaetalmaxLCWv4	1,00	10,23
	(0,00)	*(0,42)*
LCWv4max	1,00	6,81
	(0,00)	*(0,09)*
Costaetalmax	1,00	3,08
	(0,00)	*(0,41)*
LCWmax	1,00	0,04
	(0,00)	*(0,00)*

	Optimalitási pontszám	Futásidő
LCWmin	0,01808	0,15
	(0,00961)	*(0,03)*
TLCWmin	0,01446	14,56
	(0,00675)	*(0,08)*
TLCWminLCWv4	0,00580	43,06
	(0,00514)	*(9,61)*
LCWv4min	0,00533	33,55
	(0,00466)	*(7,32)*
SOmin	0,00297	375,47
	(0,00302)	*(144,71)*
Costaetalmin	0,00222	129,19
	(0,00190)	*(42,04)*
SOminLCWv4	0,00227	392,16
	(0,00269)	*(144,78)*
CostaetalminLCWv4	0,00125	147,93
	(0,00175)	*(42,19)*

7.4. Táblázat. Maximum- (fent) és minimumkereső (lent) algoritmusok eredményei, $m = 60, k = 10$. Szürkével jelölve a redundáns heurisztikákat.

8. fejezet

Összegzés

A disszertáció céljaként az m-szobatárs probléma elemzését, elméleti és gyakorlati megoldhatóságának vizsgálatát, valamint általánosan az egyenletes klaszterezési problémák áttekintését fogalmaztuk meg.

A dolgozatban az egyenletes klaszterezési problémák tekintetében a 2. Fejezetben tárgyaltuk ezek lehetséges alkalmazásainak körét.

A 3. Fejezetben bemutattuk a problémák ismert nehézségi eredményeit, majd az egyenletes MSSC probléma általánosításaként megfogalmaztuk a pMIN-mDM, valamint a pMAX-mDM problémákat, amelyekről beláttuk, hogy legalább háromfős csoportok kialakítására rendre $p \geq 1$ és $p \geq 2$ értékekre általános dimenzióban NP-nehezek.

A 4. Fejezetben formálisan megadtuk az m-szobatárs problémát, amellyel egy Pareto-hatékony megoldási koncepciót határoztunk meg. A felírás révén felállítottuk azt az elemzési keretet, amelyben egyszerre tekinthettük a feladat minimalizálási és maximalizálási verzióit. A minimalizálási feladatot a 2MIN-mDM, míg a maximalizálási feladatot a 2MAX-mDM problémák által adtuk meg, amelyből egyből következik, hogy az m-szobatárs probléma legalább 3-fős szobák esetén NP-nehéz, azaz jelenlegi ismereteink szerint nem tudjuk hatékonyan megoldani. A fejezet végén tárgyaltuk a megközelítés előnyeit és hátrányait a stabil szobatársak felíráshoz képest.

Az 5. Fejezetben áttekintettük azokat az egyenletes klaszterezési problémák során alkalmazott algoritmusokat, amelyek valamilyen garanciát adnak az általuk talált megengedett megoldás szuboptimalitására. Ezek közül az m-szobatárs feladat szempontjából a kúp optimalizálás releváns, hiszen a minimalizálási probléma esetét tárgyalja.

Az m-szobatárs probléma gyakorlati megoldásának szempontjából az irodalomban ta-

lálható legfontosabb algoritmusokat a 6. Fejezetben mutattuk be. Az eljárások köréhez két módon járultunk hozzá:

- egyrészt megfogalmaztunk 6 különböző heurisztikus eljárást a klaszterelemzéshez kapcsolódó eljárások által adott csoportok kiegyenlítésére;
- másrészt megadtuk a fuzzy c-közép egyenlő klaszterméretekkel eljárás egy előzőektől eltérő kiegyenlítését, amely az `eqFCMv2` heurisztikát eredményezte;
- harmadrészt megvizsgáltuk az `LCW` algoritmus (Weitz és Lakshminarayanan, 1996) esetén a nagyobb méretű cserék lehetőségét, és megfogalmaztunk három heurisztikus eljárást, amelyek kettes és hármas cserék segítségével keresik az optimumot.

A 7. Fejezet során végrehajtottuk az m-szobatárs probléma megoldására adott heurisztikák több lépésből álló vizsgálatát a megfogalmazott, minimum- és maximumfeladatot is tartalmazó elemzési keretben. Egyúttal magának az m-szobatárs problémának az elemzését is elvégeztük.

A vizsgálat során az egyik fő, és meglehetősen látványos eszközünk a szobabeosztásoknak az az általunk megadott konstrukciója, amely lehetővé tette, hogy alacsony hallgatói számok mellett az össztávolságokat hisztogramon ábrázoljuk. Az algoritmus felírásához a szobabeosztások formális definícióját is megadtuk.

Másik lényeges hozzájárulásunk pedig az optimalitási pontszám megadása, amelyet az elemzési keret minimum- és maximumproblémát is tartalmazó sajátossága alapján tudtunk definiálni. Ennek segítségével ábrázolni tudtuk az algoritmusok egymáshoz képest vett teljesítményét, és a tesztelt heurisztikák alkalmazásával elérhető maximum- és minimumértékek különbségéhez képest is ki tudtuk értékelni az algoritmusokat. A fejezet során megfogalmazott eredményeink a következőek:

- a megfogalmazott `eq1-eq6` kiegyenlítő eljárások közül az `eq4` kiegyenlítő módszer teljesített a legjobban, vagyis az, amelyik a legkisebb elemszámú klaszter felől indulva, és afelé csorgatja az elemeket;
- az irodalomban hagyományosan tekintett, való életből származó ‘Iris’ és ‘Seeds’ adathalmazokon a legtöbb minimalizáló eljárás hasonlóan jól teljesít, így nem lehet ez alapján különbséget tenni közöttük;
- az egyszerű konstruktív módszerek, valamint a klaszterközéppontok meghatározását és az elemek klaszterközéppontokhoz való hozzárendelését alternáló eljárások nem teljesítenek jól a feladat megoldása során;

- a kis hallgatói létszámmal rendelkező esetek a szobabeosztások költségeinek ferde eloszlását mutatják, amely a minimalizálási és maximalizálási problémák lehetséges általános aszimmetriájára utal;
- a nagy hallgatói létszám esetén elvégzett elemzés megmutatja, hogy
  - párok esetén relatíve közel tudunk kerülni a tényleges optimumokhoz, de az `LCW` eredménye és az optimum közötti távolságot a szofisztikáltabb eljárásokkal is csak részben tudjuk ledolgozni;
  - az optimalizálási problémák minimalizálási és maximalizálási feladat, valamint kis- és nagyméretű csoportok esetén is más-más karakterisztikákkal rendelkeznek, amely a heurisztikus eljárások teljesítményeiben tükröződik;
  - a minimalizálási feladat megoldásához szignifikánsan nagyobb futásidő szükséges;
  - a hármas cseréket megvalósító `LCWv4` heurisztika bizonyos esetekben önmagában is releváns lehet;
  - a hármas cseréket megvalósító `LCWv4` módszerben kezdeti értékként alkalmazva valamely másik eljárás által adott megengedett megoldást, előbbi jelentősen javíthat azok célfüggvényértékén;
  - a lokális keresést alkalmazó eljárások köréből a legegyszerűbbnek számító és egyben leggyorsabb `LCW` eljárás teljesítménye nem sokkal marad el a szofisztikáltabb módszerek eredményeitől, így a gyakorlati alkalmazás esetén a konkrét céltól függően ezzel is elfogadható megoldáshoz juthatunk.

További kutatási irányként elsőként érdemes lenne pótolni az egyenletes klaszterezés témakörében a pMAX-mDM probléma hiányzó nehézségi eredményét $p = 1$ esetén, vagyis a távolságok abszolútértékeinek összegére.

Az elemzés során láthattuk, hogy a nem redundáns heurisztikák növekvő futásidők mellett átlagosan egyre jobb célfüggvényértékeket adnak eredményül. Ugyanakkor, ahogy például Costa és szerzőtársai (2017) elemzésében is megjelenik, vizsgálhatjuk azt is, hogy egységnyi idő alatt mennyire teljesítenek jól az eljárások. Ezt két szempont miatt érdemes figyelembe venni. Egyrészt a nagyon alacsony futásidővel rendelkező heurisztikákat érdemes lehet többször egymás után futtatni, hátha így hatékonyan tudjuk növelni a relatíve rossz teljesítményt. Másrészt pedig előfordulhat, hogy egy komplex módszer egy egyszerűbb eljárás egységnyi futásideje alatt jobb eredményt talál meg, így igazából bármilyen

futásidő mellett érdemes lehet az összetettebb eljárást alkalmazni, legfeljebb hamarabb leállítjuk a futását.

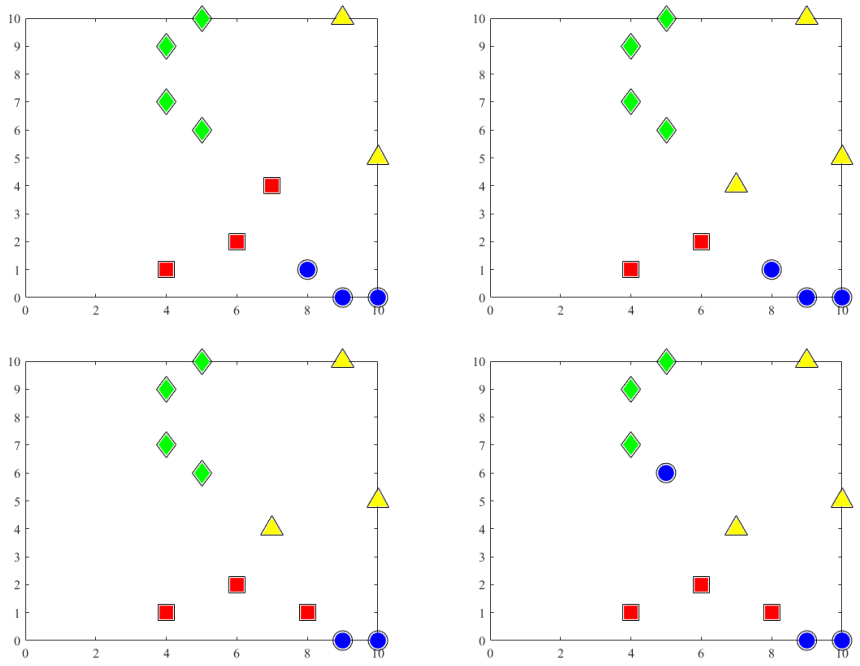
Szintén a futásidőkhöz kapcsolódóan, az elemzés során észrevehettük, hogy a hallgatók számának növelésével drasztikusan nőnek a futásidők. Emiatt a komplexebb heurisztikák egészen biztosan nem alkalmasak az adatbányászati alkalmazásoknál előforduló, akár több tízezer fős minták particionálására, mint amilyet Banerjee és Ghosh (2006) is tekintett. A szerzők által javasolt, stabil házasságokat alkalmazó eljárást érdemes lenne összevetni az általunk elvégzett elemzésben alacsony futásidőt produkáló módszerekkel, mint a hagyományos klaszterezéshez kötődő algoritmusok egy része, valamint az `LCW` eljárás.

Ezenkívül a gyakorlati alkalmazhatóság szempontjából érdemes lenne megfontolni az egyéni preferenciák alkalmazhatóságát a modellben. Például olyan esetekben, amikor kettő vagy több hallgató mindenképp ugyanabba a szobába szeretne kerülni. Ekkor egyrészt kérdés, hogy ezt hogyan lehet a modell szintjén kikötni (a 0 távolságok egy kézenfekvő megoldásnak tűnik), másrészt pedig kérdés, hogy a gyakorlati megoldás során ezt hogyan tudjuk kikényszeríteni. Hasonló megfontolások tárgya lehet, ha egyes hallgatók biztosan nem akarnak egy szobába kerülni valaki mással.

Melléklet

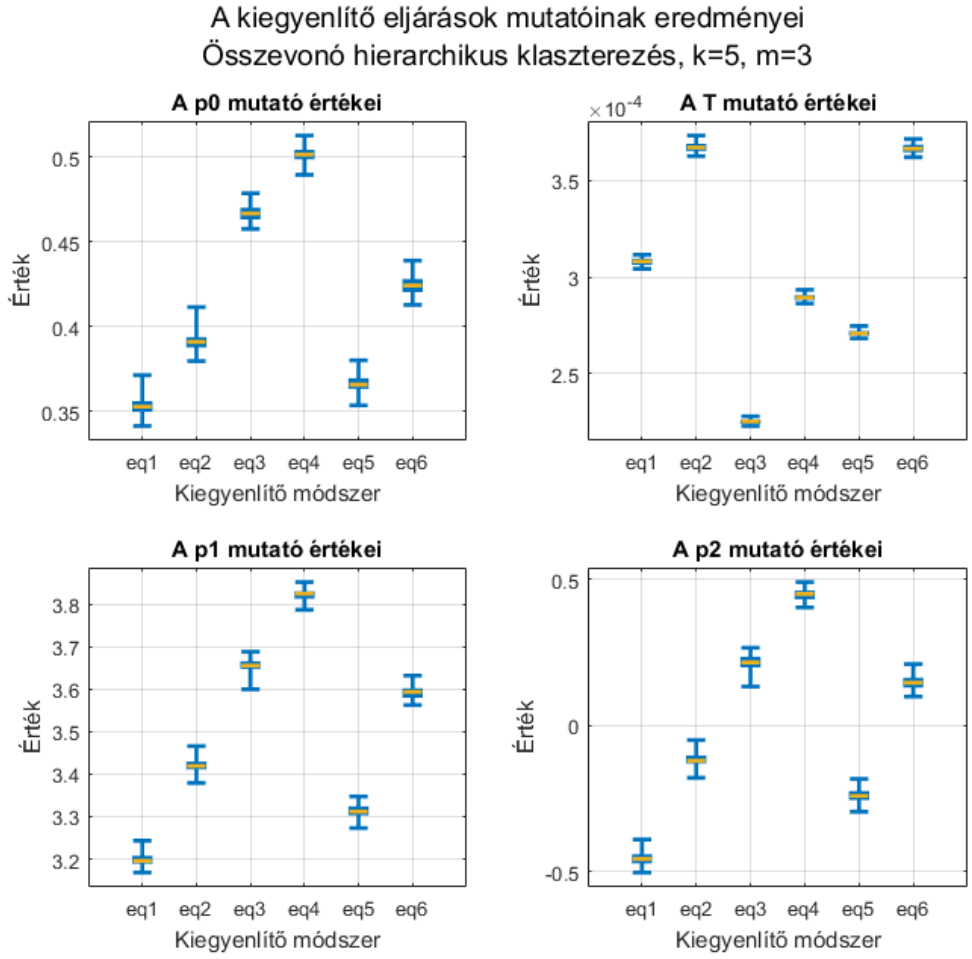
A. További ábrák

A.1. Negatív példa az eq3 kiegyenlítő módszer futására

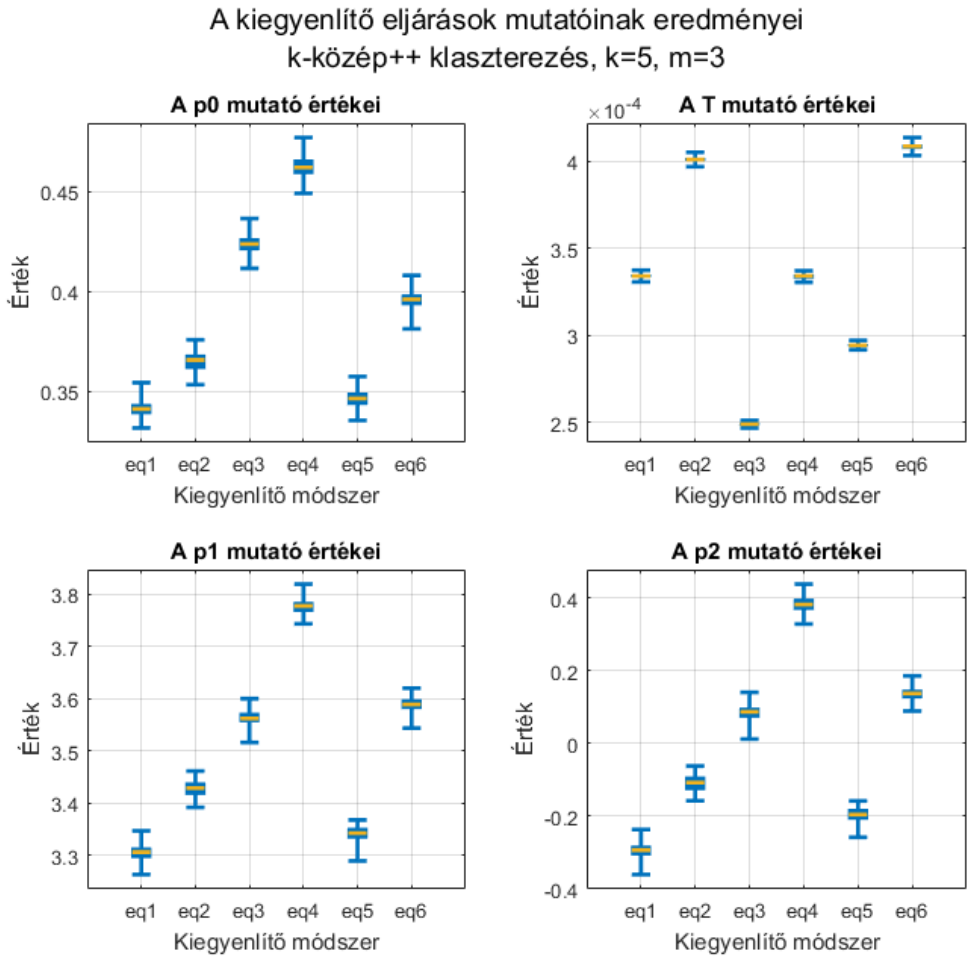


A.1. ábra. Az eq3 kiegyenlítő módszer egy futásának eredményeként kapott ábrák. Negatív példa.

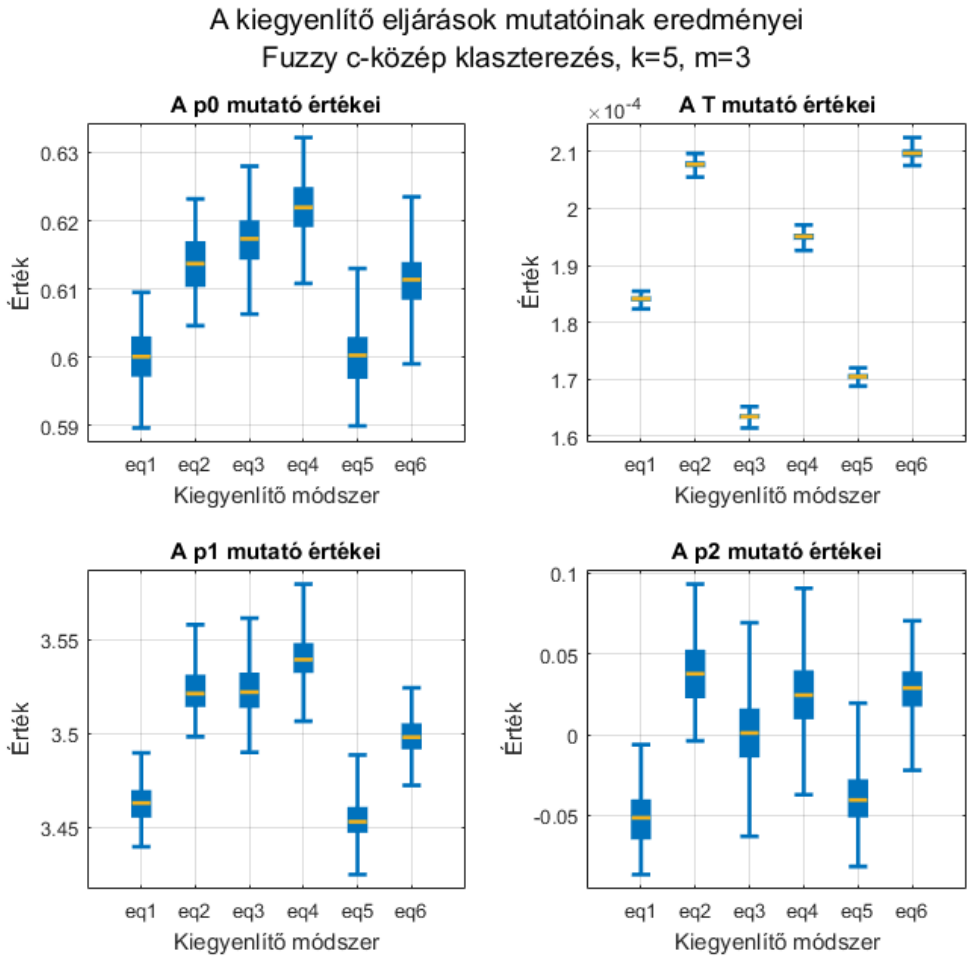
A.2. A kiegyenlítő eljárások eredményei, $k = 5, m = 3$



A.2. ábra. A kiegyenlítő eljárások (`eq1-eq6`) eredményei. Összevonó hierarchikus klaszterezés, $k = 5, m = 3$.

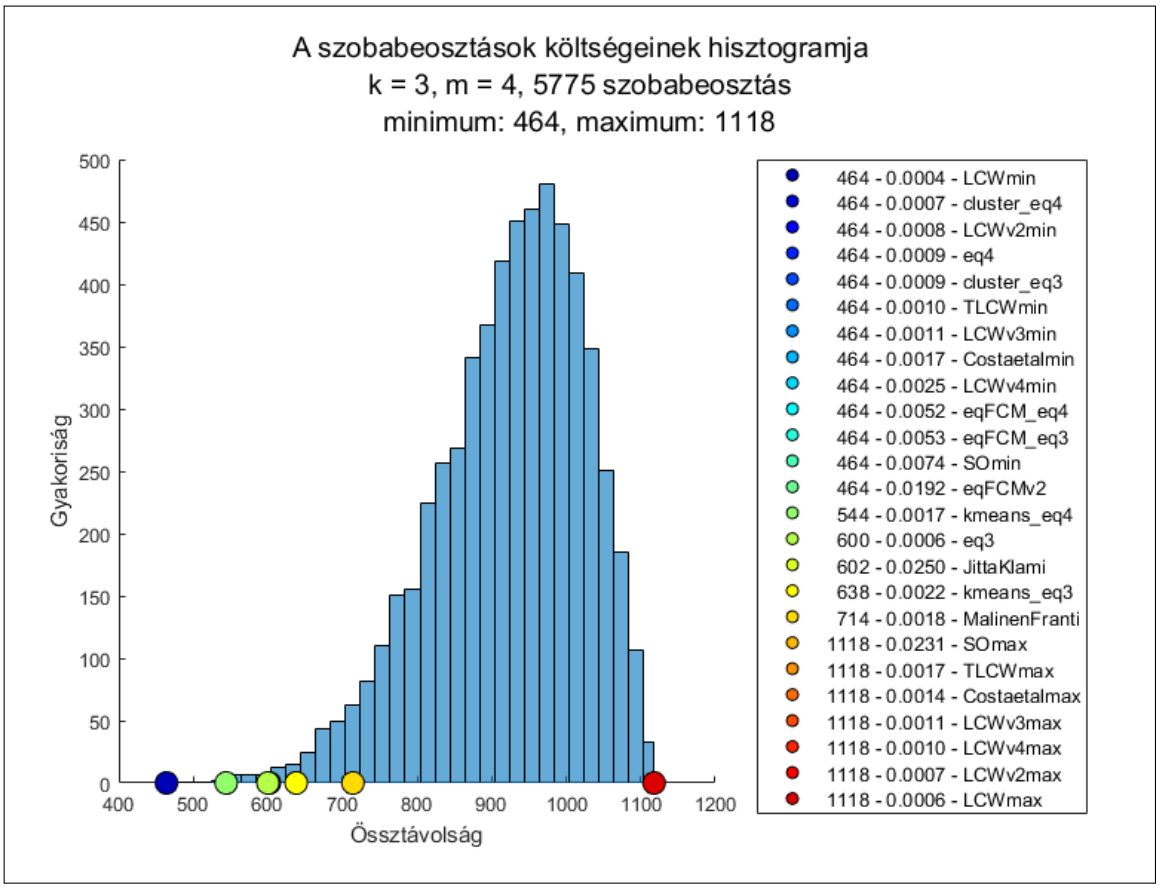


A.3. ábra. A kiegyenlítő eljárások (`eq1-eq6`) eredményei. k-közép++ klaszterezés, $k = 5, m = 3$.

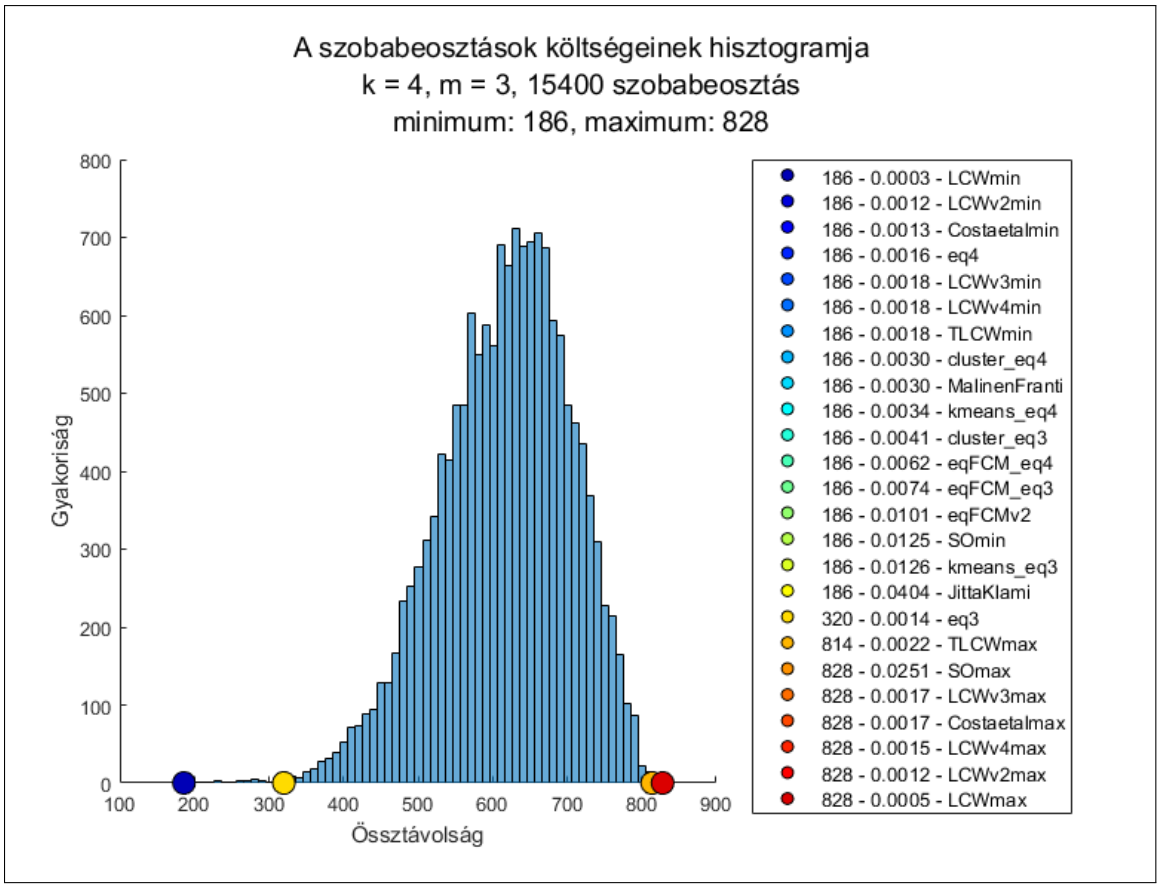


A.4. ábra. A kiegyenlítő eljárások (`eq1-eq6`) eredményei. Fuzzy c-közép klaszterezés, $k = 5, m = 3$.

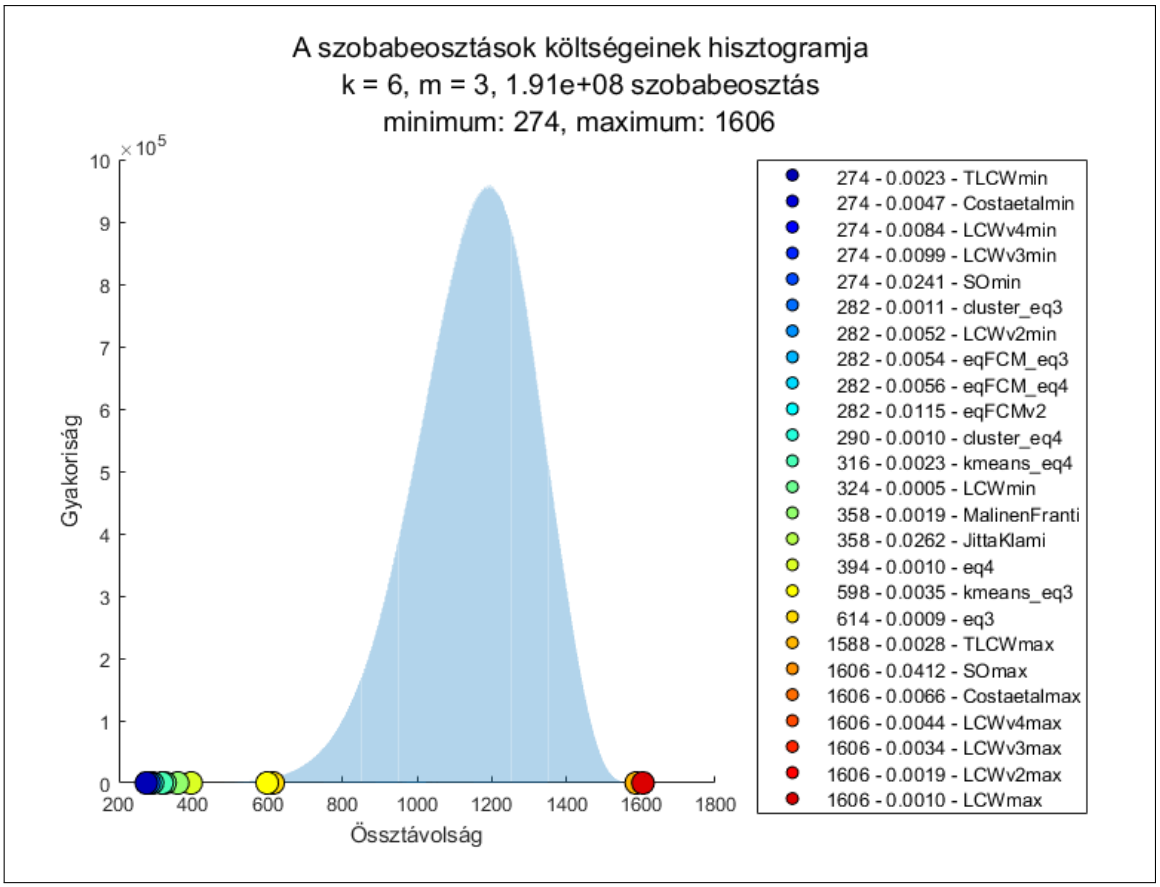
A.3. További hisztogramok kis k és kis m esetén



A.5. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 3, m = 4$, 5775 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 0,2361 mp. Ferdeség: -0,2102. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.



A.6. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 4, m = 3$, 15400 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 0,7209 mp. Ferdeség: -0,4372. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.



A.7. ábra. A szobabeosztások költségeinek hisztogramja, valamint a minimum- és maximumkereső algoritmusok futásainak eredményei. Egy szimulált eset, $k = 6, m = 3$, 1,91e+08 lehetséges szobabeosztás, a szobabeosztások generálásának és az optimumok keresésének futásideje 2 óra 11 p. Ferdeség: -0,2361. A feliratok magyarázata: célfüggvényérték - futásidő - algoritmus neve.

Irodalomjegyzék

Ageev, A. A., Kel'manov, A. V. és Pyatkin, A. V. (2014). Complexity of the weighted max-cut in Euclidean space. *Journal of Applied and Industrial Mathematics*, 8(4): 453–457. DOI: 10.1134/S1990478914040012.

Aloise, D., Deshpande, A., Hansen, P. és Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75:245—248. DOI: 10.1007/s10994-009-5103-0.

Aloise, D., Caporossi, G., Hansen, P., Liberti, L., Perron, S. és Ruiz, M. (2013). Modularity maximization in networks by variable neighborhood search. In Bader, D. A., Meyerhenke, H., Sanders, P. és Wagner, D., editors, *Graph Partitioning and Graph Clustering*, volume 588, pages 113—227. DOI: 10.1090/conm/588/11705.

Aloise, D. J., Aloise, D., Rocha, C. T. M., Ribeiro, C. C., Filho, J. C. R. és Moura, L. S. S. (2006). Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, 154(5):695–702. DOI: 10.1016/j.dam.2004.09.021.

Arani, T. és Lotfi, V. (1989). A three phased approach to final exam scheduling. *IIE Transactions*, 21:86–96. DOI: 10.1080/07408178908966211.

Arkin, E. M., Bae, S. W., Efrat, A., Okamoto, K., Mitchell, J. S. B. és Polishchuk, V. (2009). Geometric stable roommates. *Information Processing Letters*, 109(4):219–224. DOI: 10.1016/j.ipl.2008.10.003.

Arora, S. és Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press. ISBN: 978-0-521-42426-4.

Arora, S., Lund, C., Motwani, R., Sudan, M. és Szegedy, M. (1998). Proof verification

and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555. DOI: 10.1145/278298.278306.

Arthur, D. és Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035. ISBN 978-0-898716-24-5.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. és Protasi, M. (2003). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer-Verlag Berlin Heidelberg 1999. ISBN: 978-3-642-63581-6. DOI: 10.1007/978-3-642-58412-1.

Awasthi, P., Bandeira, A. S., Charikar, M., Krishnaswamy, R., Villar, S. és Ward, R. (2015). Relax, no need to round: Integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 191–200. DOI: 10.1145/2688073.2688116.

Baeza-Yates, R. és Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley, 1st edition. ISBN: 020139829X.

Baker, K. R. és Powell, S. G. (2002). Methods for assigning students to groups: A study of alternative objective functions. *Journal of the Operational Research Society*, 53(4): 397–404. DOI: 10.1057/palgrave.jors.2601307.

Banerjee, A. és Ghosh, J. (2002). On scaling up balanced clustering algorithms. In *Proceedings of the Second SIAM International Conference on Data Mining*, pages 333–349. DOI: 10.1137/1.9781611972726.20.

Banerjee, A. és Ghosh, J. (2006). Scalable clustering algorithms with balancing constraints. *Data Mining and Knowledge Discovery*, 13:365–395. DOI: 10.1007/s10618-006-0040-z.

Belacel, N., Hansen, P. és Mladenović, N. (2002). Fuzzy J-Means: a new heuristic for fuzzy clustering. *Pattern Recognition*, 35(10):2193–2200. DOI: 10.1016/S0031-3203(01)00193-5.

Benyó, B., Fék, M., Kiss, I., Kóczy, A., Kondorosi, K., Mészáros, T., Román, G., Szeberényi, I. és Sziray, J. (2000). *Operációs rendszerek mérnöki megközelítésben*. Panem Könyvkiadó. ISBN: 9635452500.

Berman, P. és Karpinski, M. (1999). On some tighter inapproximability results. In *International Colloquium on Automata, Languages, and Programming*, pages 200–209. Springer, Berlin, Heidelberg. DOI: 10.1007/3-540-48523-6_17.

Bertoni, A., Goldwurm, M., Lin, J. és Saccà, F. (2012). Size constrained distance clustering: separation properties and some complexity results. *Fundamenta Informaticae*, 115(1):125–139. DOI: 10.3233/FI-2012-644.

Bhadury, J., Mighty, E. J. és Damar, H. (2000). Maximizing workforce diversity in project teams: a network flow approach. *Omega*, 28(2):143–153. DOI: 10.1016/S0305-0483(99)00037-7.

Biró, P. (2006). Stabil párosítási modellek és ezeken alapuló központi párosító programok. *Szigma*, 37(3-4):153–175. `https://journals.lib.pte.hu/index.php/szigma/article/view/1096`.

Biró, P. és McDermid, E. (2010). Three-sided stable matchings with cyclic preferences. *Algorithmica*, 58:5–18. DOI: 10.1007/s00453-009-9315-2.

Biró, P., Iñarra, E. és Molis, E. (2016). A new solution concept for the roommate problem: $\mathcal{Q}$-stable matchings. *Mathematical Social Sciences*, 79:74–82. DOI: 10.1016/j.mathsocsci.2015.12.001.

Blum, M., Floyd, R. W., Pratt, V., Rivest, R. L. és Tarjan, R. E. (1973). Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461. DOI: 10.1016/S0022-0000(73)80033-9.

Boussaïd, I., Lepagnot, J. és Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117. DOI: 10.1016/j.ins.2013.02.041.

Bradley, P. S., Bennett, K. P. és Demiriz, A. (2000). Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research. `https://www.microsoft.com/en-us/research/publication/constrained-k-means-clustering/`.

Brimberg, J., Mladenović, N., Todosijević, R. és Urošević, D. (2017). A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. *Optimization Letters*, 11:313–327. DOI: 10.1007/s11590-015-0973-5.

Burkard, R. E. (2013). Quadratic assignment problems. In Pardalos, P. M., Du, D.-Z. és Graham, R. L., editors, *Handbook of Combinatorial Optimization*, pages 2741–2814. Springer, New York, NY. DOI: 10.1007/978-1-4419-7997-1_22.

Caprara, A. és Rizzi, R. (2002). Packing triangles in bounded degree graphs. *Information Processing Letters*, 84(4):175–180. DOI: 10.1016/S0020-0190(02)00274-0.

Chang, T.-C. és Wysk, R. A. (1985). *An Introduction to Automated Process Planning Systems*. Prentice-Hall International Series in Industrial and Systems Engineering. Prentice-Hall, Inc., Englewood Cliffs, New Jersey. ISBN: 0134781406.

Chataigner, F., Manić, G., Wakabayashi, Y. és Yuster, R. (2009). Approximation algorithms and hardness results for the clique packing problem. *Discrete Applied Mathematics*, 157(7):1396–1406. DOI: 10.1016/j.dam.2008.10.017.

Chen, J. és Roy, S. (2021). Euclidean 3D stable roommates is NP-hard. Technical report. `https://arxiv.org/abs/2108.03868`.

Chen, Y., Chen, Z.-Z., Lin, G., Wang, L. és Zhang, A. (2021). A randomized approximation algorithm for metric triangle packing. *Journal of Combinatorial Optimization*, 41: 12–27. DOI: 10.1007/s10878-020-00660-7.

Chen, Z.-Z., Tanahashi, R. és Wang, L. (2009). An improved randomized approximation algorithm for maximum triangle packing. *Discrete Applied Mathematics*, 157(7):1640–1646. DOI: 10.1016/j.dam.2008.11.009.

Chen, Z.-Z., Tanahashi, R. és Wang, L. (2010). Erratum to “An improved randomized approximation algorithm for maximum triangle packing" [Discrete Applied Mathematics 157 (2009) 1640–1646]. *Discrete Applied Mathematics*, 158(9):1045–1047. DOI: 10.1016/j.dam.2010.01.011.

Chung, K.-S. (2000). On the existence of stable roommate matchings. *Games and Economic Behavior*, 33(2):206–230. DOI: 10.1006/game.1999.0779.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. DOI: 10.1145/800157.805047.

Costa, L. R., Aloise, D. és Mladenović, N. (2017). Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. *Information Sciences*, 415-416:247–253. DOI: 10.1016/j.ins.2017.06.019.

Dasgupta, S. (2008). The hardness of k-means clustering. Technical report, Department of Computer Science and Engineering, University of California, San Diego. `http://cseweb.ucsd.edu/~dasgupta/papers/kmeans.pdf`.

Deineko, V. G. és Woeginger, G. J. (2013). Two hardness results for core stability in hedonic coalition formation games. *Discrete Applied Mathematics*, 161(13-14):1837–1842. DOI: 10.1016/j.dam.2013.03.003.

Drineas, P., Frieze, A., Kannan, R., Vempala, S. és Vinay, V. (2004). Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33. DOI: 10.1023/B:MACH.0000033113.59016.96.

Edmonds, J. (1965a). Paths, trees, and flowers. *Canadian Journal of mathematics*, 17: 449–467. DOI: 10.4153/CJM-1965-045-4.

Edmonds, J. (1965b). Maximum matching and a polyhedron with $0,1$-vertices. *Journal of Research of the National Bureau of Standards*, 69B(1-2):125–130. DOI: 10.6028/jres.069b.013.

Edwards, A. W. F. és Cavalli-Sforza, L. L. (1965). A method for cluster analysis. *Biometrics*, 21(2):362–375. DOI: 10.2307/2528096.

Eschenauer, L. és Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. DOI: 10.1145/586110.586117.

Fagin, R. (1974). Generalized first-order spectra, and polynomial-time recognizable sets. In *Complexity of computation, SIAM-AMS Proceedings*, volume 7, pages 43–73. `https://researcher.watson.ibm.com/researcher/files/us-fagin/genspec.pdf`.

Fan, Z.-P., Chen, Y., Ma, J. és Zeng, S. (2011). A hybrid genetic algorithmic approach to the maximally diverse grouping problem. *Journal of the Operational Research Society*, 62(1):92–99. DOI: 10.1057/jors.2009.168.

Fasulo, D. (1999). An analysis of recent work on clustering algorithms. Technical report, Department of Computer Science & Engineering, University of Washington. URL: `https://dada.cs.washington.edu/research/tr/2001/03/UW-CSE-01-03-02.pdf`.

Feo, T. A. és Khellaf, M. (1990). A class of bounded approximation algorithms for graph partitioning. *Networks*, 20(2):181–195. DOI: 10.1002/net.3230200205.

Feo, T. A., Goldschmidt, O. és Khellaf, M. (1992). One-half approximation algorithms for the k-partition problem. *Operations Research*, 40:S170–S173. `https://www.jstor.org/stable/3840846`.

Ferenczi, M. (2014). *Matematikai logika*. Műszaki Könyvkiadó, Budapest, 2. kiadás. ISBN: 978-963-16-2870-8.

Fisher, W. D. (1958). On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798. DOI: 10.1080/01621459.1958.10501479.

Gabow, H. N. (1976). An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23(2):221–234. DOI: 10.1145/321941.321942.

Gabow, H. N. (1990). Data structures for weighted matching and nearest common ancestors with linking. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. `https://dl.acm.org/doi/10.5555/320176.320229`.

Gale, D. és Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15. DOI: 10.2307/2312726.

Gallego, M., Laguna, M., Marti, R. és Duarte, A. (2013). Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, 64(5):724–734. DOI: 10.1057/jors.2012.66.

Ganganath, N., Cheng, C.-T. és Tse, C. K. (2014). Data clustering with cluster size constraints using a modified k-means algorithm. In *International Conference on Cyber-*

Enabled Distributed Computing and Knowledge Discovery, Shanghai, China, 2014, pages 158–161. DOI: 10.1109/CyberC.2014.36.

Garey, M. R. és Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman, 1st edition. ISBN: 0-7167-1044-7.

Garey, M. R., Johnson, D. S. és Stockmeyer, L. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267. DOI: 10.1016/0304-3975(76)90059-1.

Ghiasi, S., Srivastava, A., Yang, X. és Sarrafzadeh, M. (2002). Optimal energy aware clustering in sensor networks. *Sensors*, 2(7):258–269. DOI: 10.3390/s20700258.

Ghosh, J. (2003). Scalable clustering. In Ye, N., editor, *The Handbook of Data Mining*, Human Factors and Ergonomics, pages 247–277. Lawrence Erlbaum Associates, Publishers. ISBN: 0-8058-4081-8.

Glover, F. W. és Laguna, M. (1997). Tabu search. Springer US. DOI: 10.1007/978-1-4615-6089-0. ISBN: 978-0-7923-8187-7.

Gupta, G. és Younis, M. (2003). Load-balanced clustering of wireless sensor networks. In *IEEE International Conference on Communications, 2003. ICC '03*, volume 3, pages 1848–1852. DOI: 10.1109/ICC.2003.1203919.

Gupta, G. K. és Ghosh, J. (2001). Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. DOI: 10.1137/1.9781611972719.8.

Guttmann-Beck, N. és Hassin, R. (1998). Approximation algorithms for min-sum p-clustering. *Discrete Applied Mathematics*, 89(1-3):125–142. DOI: 10.1016/S0166-218X(98)00100-0.

Han, J., Kamber, M. és Tung, A. K. H. (2001). Spatial clustering methods in data mining: A survey. In Miller, H. J. és Han, J., editors, *Geographic Data Mining and Knowledge Discovery*, chapter 8th, pages 201–231. Taylor & Francis. ISBN: 0-415-23369-0. URL: `https://www.comp.nus.edu.sg/~atung/publication/gkdbk01.pdf`.

Hansen, P. és Mladenović, N. (2001a). J-Means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2):405–413. DOI: 10.1016/S0031-3203(99)00216-2.

Hansen, P. és Mladenović, N. (2001b). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467. DOI: 10.1016/S0377-2217(00)00100-4.

Hansen, P. és Mladenović, N. (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802–817. DOI: 10.1016/j.dam.2005.05.020.

Hansen, P., Ruiz, M. és Aloise, D. (2012). A VNS heuristic for escaping local extrema entrapment in normalized cut clustering. *Pattern Recognition*, 45(12):4337–4345. DOI: 10.1016/j.patcog.2012.04.029.

Hassin, R. és Rubinstein, S. (2006a). An approximation algorithm for maximum triangle packing. *Discrete Applied Mathematics*, 154(6):971–979. DOI: 10.1016/j.dam.2005.11.003.

Hassin, R. és Rubinstein, S. (2006b). Erratum to "An approximation algorithm for maximum triangle packing" : [Discrete Applied Mathematics 154 (2006) 971–979]. *Discrete Applied Mathematics*, 154(18):2620. DOI: 10.1016/j.dam.2006.05.005.

Hassin, R. és Rubinstein, S. (2006c). An improved approximation algorithm for the metric maximum clustering problem with given cluster sizes. *Information Processing Letters*, 98(3):92–95. DOI: 10.1016/j.ipl.2005.12.002.

Hassin, R., Rubinstein, S. és Tamir, A. (1997). Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21(3):133–137. DOI: 10.1016/S0167-6377(97)00034-5.

Hettich, S. és Pazzani, M. J. (2006). Mining for proposal reviewers: lessons learned at the National Science Foundation. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 862–871. Philadelphia, PA, USA. DOI: 10.1145/1150402.1150521.

Huang, C.-C. (2007). Two's company, three's a crowd: Stable family and threesome roommates problems. In *Algorithms – ESA 2007*, Lecture Notes in Computer Science,

vol 4698, pages 558–569. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-75520-3_50.

Hulett, H., Will, T. G. és Woeginger, G. J. (2008). Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596. DOI: 10.1016/j.orl.2008.05.004.

Hurkens, C. A. J. és Schrijver, A. (1989). On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72. DOI: 10.1137/0402008.

Höppner, F. és Klawonn, F. (2008). Clustering with size constraints. In *Computational Intelligence Paradigms*, volume 137 of *Studies in Computational Intelligence*, pages 167–180. ISBN 978-3-540-79473-8. DOI: 10.1007/978-3-540-79474-5_8.

Irving, R. W. (1985). An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595. DOI: 10.1016/0196-6774(85)90033-1.

Irving, R. W. és Manlove, D. F. (2002). The stable roommates problem with ties. *Journal of Algorithms*, 43(1):85–105. DOI: 10.1006/jagm.2002.1219.

Iwama, K., Miyazaki, S. és Okamoto, K. (2007). Stable roommates problem with triple rooms. In *10th Korea–Japan Joint Workshop on Algorithms and Computation*. `http://www.researchgate.net/publication/228816339_Stable_roommates_problem_with_triple_rooms`.

Jain, A. K., Murty, M. N. és Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323. DOI: 10.1145/331499.331504.

Jitta, A. és Klami, A. (2018). On controlling the size of clusters in probabilistic clustering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pages 3350–3357. Palo Alto, CA: AAAI Press. `https://ojs.aaai.org/index.php/AAAI/article/view/11793`.

Kann, V. (1991). Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35. DOI: 10.1016/0020-0190(91)90246-E.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer, Boston, MA. DOI: 10.1007/978-1-4684-2001-2_9.

Karp, R. M. (1975). On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68. DOI: 10.1002/net.1975.5.1.45.

Katona, Gy. Y., Recski, A. és Szabó, C. (2006). *A számítástudomány alapjai*. Typotex, 5th edition. ISBN: 963 9664 19 7.

Kel'manov, A. V. és Pyatkin, A. V. E. (2016). On the complexity of some quadratic Euclidean 2-clustering problems. *Computational Mathematics and Mathematical Physics*, 56:491–497. DOI: 10.1134/S096554251603009X.

King, J. R. és Nakornchai, V. (1982). Machine-component group formation in group technology: review and extension. *International Journal of Production Research*, 20 (2):117–133. DOI: 10.1080/00207548208947754.

Kirkpatrick, D. G. és Hell, P. (1978). On the completeness of a generalized matching problem. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 240–245. DOI: 10.1145/800133.804353.

Kirkpatrick, D. G. és Hell, P. (1983). On the complexity of general graph factor problems. *SIAM Journal on Computing*, 12(3):601–609. DOI: 10.1137/0212040.

Király, B. és Tóth, L. (2011). Kombinatorika jegyzet és feladatgyűjtemény. Pécsi Tudományegyetem.

Knuth, D. E. (1977). *Stable Marriage and Its Relation to Other Combinatorial Problems*, volume 10 of *CRM Proceedings and Lecture Notes*. American Mathematical Society. Fordította Goldstein, M., eredeti: Knuth, D. E. (1976) *Mariages Stables*. Montréal: Les Presses de l'Université de Montréal. ISBN: 978-0-8218-0603-6. DOI: 10.1090/crmp/010.

Kondor, G. (2018). k-szobatárs probléma metrikus térben – klaszterezés egyenlő elemszámú és kisméretű csoportokkal. In *Tavaszi Szél 2018 Konferencia = Spring Wind 2018: Konferenciakötet II.*, pages 549–563. ISBN: 9786155586316.

Kondor, G. (2022a). NP-hardness of m-dimensional matching problems. *Műhelytanulmány.*

Kondor, G. (2022b). Egyoldali párosítási piacok nehézségi eredményei magasabb dimenzióban. *Közgazdasági Szemle. Megjelenés alatt.*

Dr. Kovács, E., Szüle, B., Fliszár, V. és Vékás, P. (2011). *Pénzügyi adatok statisztikai elemzése: Egyetemi tankönyv.* Tanszék Kft., Budapest.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97. DOI: 10.1002/nav.3800020109.

Kumar, K. R., Kusiak, A. és Vannelli, A. (1986). Grouping parts and components in flexible manufacturing system. *European Journal of Operational Research*, 24(3):387–397. DOI: 10.1016/0377-2217(86)90032-9.

Lam, C.-K. és Plaxton, C. G. (2019). On the existence of three-dimensional stable matchings with cyclic preferences. In Fotakis, D. és Markakis, E., editors, *Algorithmic Game Theory*, SAGT 2019. Lecture Notes in Computer Science, vol 11801. Springer, Cham. DOI: 10.1007/978-3-030-30473-7_22.

Lan, Y., Xiuli, C. és Meng, W. (2009). An energy-balanced clustering routing algorithm for wireless sensor networks. In *2009 WRI World Congress on Computer Science and Information Engineering*, pages 316–320. IEEE. DOI: 10.1109/CSIE.2009.559.

Levin, L. A. (1973). Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266. `http://www.mathnet.ru/eng/ppi914`.

Liao, Y., Qi, H. és Li, W. (2013). Load-balanced clustering algorithm with distributed self-organization for wireless sensor networks. *IEEE Sensors Journal*, 13(5):1498–1506. DOI: 10.1109/JSEN.2012.2227704.

Lichtenstein, D. (1982). Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343. DOI: 10.1137/0211025.

Lin, J., Bertoni, A. és Goldwurm, M. (2016). Exact algorithms for size constrained 2-clustering in the plane. *Theoretical Computer Science*, 629:80–95. DOI: 10.1016/j.tcs.2015.10.005.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137. DOI: 10.1109/TIT.1982.1056489.

Lotfi, V. és Cerveny, R. (1991). A final-exam-scheduling package. *Journal of the Operational Research Society*, 42(3):205–216. DOI: 10.1038/sj/jors/0420302.

Lovász, L. és Plummer, M. (1986). *Matching Theory*. Akadémiai Kiadó, Budapest. ISBN: 963-05-4168-8.

Lynch, P. J. és Horton, S. (1999). *Web Style Guide: Basic Design Principles for Creating Web Sites*. Yale University Press. ISBN: 0300076754.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. `http://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf`.

Mahajan, M., Nimbhorkar, P. és Varadarajan, K. (2009). The planar k-means problem is NP-hard. In *WALCOM: Algorithms and Computation*, pages 274–285. WALCOM 2009. Lecture Notes in Computer Science, vol 5431. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-00202-1_24.

Malinen, M. I. és Fränti, P. (2014). Balanced k-means for clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, S+SSPR 2014, pages 32–41. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-662-44415-3_4.

McLachlan, G. J. és Peel, D. (2000). *Finite mixture models*. John Wiley & Sons. ISBN: 978-0-471-00626-8, DOI: 10.1002/0471721182.

Mingers, J. és O'Brien, F. A. (1995). Creating student groups with similar characteristics: A heuristic approach. *Omega*, 23(3):313–321. DOI: 10.1016/0305-0483(95)00014-F.

Mladenović, N., Todosijević, R. és Urošević, D. (2016). Less is more: Basic variable neighborhood search for minimum differential dispersion problem. *Information Sciences*, 326:160–171. DOI: 10.1016/j.ins.2015.07.044.

Morrill, T. (2010). The roommates problem revisited. *Journal of Economic Theory*, 145 (5):1739–1756. DOI: 10.1016/j.jet.2010.02.003.

Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R. és Parthiban, P. (2009). Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science*, 8(4):480–487. `https://mathscinet.ams.org/mathscinet-getitem?mr=2595468`.

Ng, C. és Hirschberg, D. S. (1991). Three-dimensional stable matching problems. *SIAM Journal on Discrete Mathematics*, 4(2):245–252. DOI: 10.1137/0404023.

Nielsen Marketing Research. (1993). *Category Management: Positioning Your Organization to Win*. McGraw-Hill. ISBN: 9780844234892.

Nobel Prize. (2012a). Press release. NobelPrize.org. `https://www.nobelprize.org/prizes/economic-sciences/2012/press-release/`.

Nobel Prize. (2012b). Scientific background. NobelPrize.org. `https://www.nobelprize.org/uploads/2018/06/advanced-economicsciences2012.pdf`.

Novick, B. (2009). Norm statistics and the complexity of clustering problems. *Discrete Applied Mathematics*, 157(8):1831–1839. DOI: 10.1016/j.dam.2009.01.003.

Papadimitriou, C. H. és Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, N.J. ISBN: 0131524623.

Papadimitriou, C. H. és Yannakakis, M. (1991). Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440. DOI: 10.1016/0022-0000(91)90023-X.

Peng, J. és Wei, Y. (2007). Approximating K-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18(1):186–205. DOI: 10.1137/050641983.

Pyatkin, A., Aloise, D. és Mladenović, N. (2017). NP-hardness of balanced minimum sum-of-squares clustering. *Pattern Recognition Letters*, 97:44–45. DOI: 10.1016/j.patrec.2017.05.033.

Ribeiro, C. C., Aloise, D., Noronha, T. F., Rocha, C. és Urrutia, S. (2008). An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. *European Journal of Operational Research*, 191(3):596–611. DOI: 10.1016/j.ejor.2007.02.003.

Ronn, E. (1990). NP-complete stable matching problems. *Journal of Algorithms*, 11(2): 285–304. DOI: 10.1016/0196-6774(90)90007-2.

Rujeerapaiboon, N., Schindler, K., Kuhn, D. és Wiesemann, W. (2019). Size matters: Cardinality-constrained clustering and outlier detection via conic optimization. *SIAM Journal on Optimization*, 29(2):1211–1239. DOI: 10.1137/17M1150670.

Santi, É., Aloise, D. és Blanchard, S. J. (2016). A model for clustering data from heterogeneous dissimilarities. *European Journal of Operational Research*, 253(3):659–672. DOI: 10.1016/j.ejor.2016.03.033.

Saunders, D. (2022). Weighted maximum matching in general graphs. MATLAB Central File Exchange. Letöltve: 2022. február 22. `https://www.mathworks.com/matlabcentral/fileexchange/42827-weighted-maximum-matching-in-general-graphs`.

Segev, D. L., Gentry, S. E., Warren, D. S., Reeb, B. és Montgomery, R. A. (2005). Kidney paired donation and optimizing the use of live donor organs. *Journal of the American Medical Association*, 293(15):1883–1890. DOI: 10.1001/jama.293.15.1883.

Simon, A. (n.d.). Matematikai logika. Egyetemi jegyzet. `http://math.bme.hu/~asimon/log.pdf`.

Späth, H. (1980). *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*. Ellis Horwood Series in Computers and Their Applications. Ellis Horwood, 1980. ISBN: 978-0853121411.

Steinhaus, H. (1956). Sur la division des corps matériels en parties. (french). *Bulletin L'Académie Polonaise des Science, Classe III*, 4:801–804. `https://mathscinet.ams.org/mathscinet-getitem?mr=90073`.

Strehl, A. és Ghosh, J. (2003). Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230. DOI: 10.1287/ijoc.15.2.208.14448.

Su, W., Hu, J., Lin, C. és Shen, S. (2015). SLA-aware tenant placement and dynamic resource provision in SaaS. In *2015 IEEE International Conference on Web Services*, pages 615–622. DOI: 10.1109/ICWS.2015.87.

Tan, J. J. M. (1991). A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12(1):154–178. DOI: 10.1016/0196-6774(91)90028-W.

Tokuyama, T. és Nakano, J. (1991). Geometric algorithms for a minimum cost assignment problem. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, SCG '91, pages 262–271. DOI: 10.1145/109648.109678.

Tung, A. K. H., Han, J., Lakshmanan, L. V. és Ng, R. T. (2001). Constraint-based clustering in large databases. In *Database Theory — ICDT 2001*, ICDT '01. Lecture Notes in Computer Science, vol 1973, pages 405–419. Springer, Berlin, Heidelberg. DOI: 10.1007/3-540-44503-X_26.

Van Leeuwen, E. J. és Van Leeuwen, J. (2012). Structure of polynomial-time approximation. *Theory of Computing Systems*, 50:641–674. DOI: 10.1007/s00224-011-9366-z.

Van Zuylen, A. (2013). Deterministic approximation algorithms for the maximum traveling salesman and maximum triangle packing problems. *Discrete Applied Mathematics*, 161(13-14):2142–2157. DOI: 10.1016/j.dam.2013.03.001.

Wagstaff, K. és Cardie, C. (2000). Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence*, ICML '00, pages 1103–1110. `https://dl.acm.org/doi/10.5555/645529.658275`.

Weitz, R. R. és Jelassi, M. T. (1992). Assigning students to groups: A multi-criteria decision support system approach. *Decision Sciences*, 23(3):746–757. DOI: 10.1111/j.1540-5915.1992.tb00415.x.

Weitz, R. R. és Lakshminarayanan, S. (1996). On a heuristic for the final exam scheduling problem. *Journal of the Operational Research Society*, 47(4):599–600. DOI: 10.1057/jors.1996.72.

Weitz, R. R. és Lakshminarayanan, S. (1998). An empirical comparison of heuristic methods for creating maximally diverse groups. *Journal of the Operational Research Society*, 49(6):635–646. DOI: 10.1057/palgrave.jors.2600510.

Yang, Y. és Padmanabhan, B. (2003). Segmenting customer transactions using a pattern-based clustering approach. In *Third IEEE International Conference on Data Mining*, pages 411–418. DOI: 10.1109/ICDM.2003.1250947.

Zhong, S. és Ghosh, J. (2003a). A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037. `https://dl.acm.org/doi/10.5555/945365.964287`.

Zhong, S. és Ghosh, J. (2003b). Scalable, balanced model-based clustering. In *Proceedings of the 2003 SIAM International Conference on Data Mining (SDM)*, pages 71–82. DOI: 10.1137/1.9781611972733.7.

Zhu, S., Wang, D. és Li, T. (2010). Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889. DOI: 10.1016/j.knosys.2010.06.003.