Hálózatok rekonstrukciója és felhasználásuk az optimalizálásban

Doktori disszertáció

Homolya Viktor

Témavezető: Dr. Vinkó Tamás

Informatika Doktori Iskola

Informatikai Intézet

Természettudományi és Informatikai Kar

Szegedi Tudományegyetem

Szeged
2024

Tartalomjegyzék

1. fejezet

Bevezetés

Az optimalizálás története kiterjedt és számos területet felölel, beleértve a matematikát, a mérnöki tudományokat, a közgazdaságtant és a számítástechnikát. Az optimalizálás célja, egy probléma lehetséges megoldásainak halmazából megtalálni a legjobb megoldást. Az optimalizálás gyökerei a matematika megjelenésével köthető össze. Az ókori görögök geometriai problémák legjobb megoldásait kutatták. Formális matematikai alapjai viszont csak a 17. században lett Newton és Leibniz munkásságának köszönhetően. Számos matematikus adott további fejlesztést ennek a területnek. A második világháború során fogalmazódott meg az operációkutatás, melyben az optimalizálás eszközeit használják gazdasági, technológiai és egyéb területek valós problémáinak megoldására. A számítógépek megjelenésével és fejlődésével új perspektívákat találtak ki az optimalizálásban, sztochasztikus optimalizálás, globális optimalizálás, heurisztikák.

A gráfelmélet és a hálózattudomány kezdetét Leonhard Euler königsbergi hidak problémájának megoldásától számítjuk. Az évek során hatalmas fejlődésen esett át köszönhetően más tudomány területekben való felhasználhatóságának. A számítógépek nagy és gyors adatfeldolgozó képességei miatt a hatalmas és komplex hálózatok elemzése lehetségessé vált. A gráfelmélet eszközeit alkalmazták szociális, közlekedési és biológiai hálózatokra. Megfogalmaztak különböző gráf tulajdonságokat mint a kisvilág gráfok, skálafüggetlen hálózatok. Ma a gráfelmélet és a hálózattudomány rendkívül interdiszciplináris terület. Matematika, számítástechnika, fizika, biológia és más területek kutatói együttműködnek a komplex rendszerek és hálózatok tanulmányozása és megértése érdekében.

1.1. A szerző hozzájárulásai

2. fejezet: A disszertáció szerzője alkotta meg és implementálta a fejezetben ismertett algoritmusokat. A tesztelések megtervezése és kiértékelése, illetve a [34] cikk a témavezetővel közös munka eredménye.

3. fejezet: A disszertáció szerzője fogalmazta meg a fejezetben ismertetett feltételeket és dolgozta ki azok bizonyításait.

4. fejezet: A fejezetben bemutatott algoritmusok implementálását, az utolsó alfejezetben szereplő tesztekhez szükséges kiegészítések kivételével, és a tesztek elvégzését a szerző hajtotta végre.

A tesztek tervezése, az eredmények értelmezése, a fejezetben bemutatott Above-Below szabály és a [31] és [33] cikkek a témavezető és a szerző közös munkája.

4. fejezet: A fejezetben ismertetett hegymászó-jellegű algoritmus megalkotása, a szakirodalomban ismert terjedési modell kivételével, a témavezető és a szerző közös munkája, a speciális Lokális Optimumok Hálózatának megfogalmazásával, a tesztek tervezésével és az eredmények értelmezésével együtt. A fejezet a [32] cikkből készült, ami a témavezető és a szerző közös eredménye.

A szerző végezte el az implementálásokat, teszteléseket, a VSD megfogalmazását és bizonyítását arról, hogy valódi távolság függvény.

1.2. Definíciók

1.2.1. Gráf fogalmak

Gráf: Egy gráfot $G = (V, E)$ jelöl, ahol V egy halmaz, melynek elemeit hívjuk csúcsoknak, és E a csúcspárok egy halmaza, melynek elemeit hívjuk éleknek. A $v_1, v_2 \in \in V$ csúcsok szomszédosak ha $(v_1, v_2) \in E$.

Egyszerű gráf: A G gráfot egyszerű gráfnak hívjuk, ha nincsenek párhuzamos élei (egy csúcspár között legfeljebb csak egy él szerepel) és hurok élei (bármely él végpontjai különböző csúcsok).

Súlyozott gráf: A $G(V, E, W)$ egy súlyozott gráf, ahol $W : E \to \mathrm{R}_+$ egy súlyfüggvény.

Összefüggő gráf: A G gráfot összefüggőnek nevezzük, ha bármely két csúcsa között létezik út.

Fa: Egy gráfot fának hívunk, ha bármely két csúcsot pontosan egy út köt össze. Ebből következik, hogy fában bármely út egyben legrövidebb út is.

Levél: Levélnek hívjuk azt a csúcsot, melynek pontosan egy szomszédja van.

Irányított gráf: Egy G gráfban az élekhez irányítottságot rendelünk, így minden irányított élhez tartozik egy kezdő- és egy végpont.

1.2.2. Utak

Út: Egy gráfban útnak hívjuk azt az egymáshoz csatlakozó élek sorozatát, melyben egy csúcs nem fordulhat elő egynél többször.

Út hossza: Egy út hossza megegyezik a benne szereplő élek számával.

Legrövidebb út: Adott u és v csúcsok közötti összes út, a legrövidebb út az az út, amely a legkevesebb éllel rendelkezik. A legrövidebb út hosszát (u és v közötti távolság) $d_G(u, v)$-vel jelöljük.

1.2.3. Gráf mértékek

Gráf mérete: A $G = (V, E)$ gráfban szereplő csúcsok száma. Jelölésben $N = |V|$.

Gráf átmérője: A $G = (V, E)$ gráf átmérője a legrövidebb utak hosszának maximuma. Jelölés: $d(G)$.

Gráf sűrűsége: A $G = (V, E)$ gráfban az élek lehetséges maximális számának és a tényleges számának az arányát jelöli.

1.2.4. Gráf reprezentációk

Szomszédsági mátrix: Egy véges irányított vagy irányítatlan N csúcsú G gráf szomszédsági mátrixa az az $N \times N$-es A mátrix, amelynek az a_{ij} eleme az i csúcsból a j csúcsba vezető élek száma. Irányítatlan esetben a főátlóban helyezkedő a_{ii} elem az i csúcs hurokéleinek számának kétszerese. Élsúlyozott gráfok esetén az élek száma helyett ugyanezen élek súlyainak összege.

Távolság mátrix: Egy véges irányított vagy irányítatlan N csúcsú G gráf távolság mátrixa az az $N \times N$-es D mátrix, amelynek a d_{ij} eleme az i csúcsból a j csúcsba vezető legrövidebb út hossza. A főátlóbeli a_{ii} elem alkalmazástól függően lehet 0 hurok létezésétől függetlenül. Élsúlyozott gráfok esetén a legrövidebb össz súlyú útban szereplő élek súlyainak összege.

1.2.5. Központiság

Centralitási mérték: A gráfok csúcs centralitása (központisága) egy olyan fogalom, mellyel a csúcsok fontosságát állapíthatjuk meg egy gráfon belül. Általánosan egy függvény, mely nemnegatív számot rendel minden egyes csúcshoz.

Fokszám centralitás - Degree Centrality: Egy $v \in V$ csúcs fokszáma (vagy foka) a hozzá kapcsolódó élek számával egyezik meg. Jelölés: $deg(v)$. Irányított gráfok

esetén megkülönböztethető a bemenő és kimenő fokszám (befok, kifok). Jelölés: $deg_+(v), deg_-(v)$. Egy csúcs fokszám centralitása megegyezik a csúcs fokszámával.

$$DC(v) = deg(v). \tag{1.1}$$

A normalizált verzióban a gráf méretével (N) osztjuk az értéket.

Közöttiség centralitás - Betweenness Centrality: Egy csúcs közöttiség centralitása (röviden és a továbbiakban BC), azzal egyezik meg, hogy a csúcs hányszor szerepelt a gráf más csúcsai között húzódó legrövidebb útjain [22]. Formálisan:

$$BC(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{1.2}$$

ahol σ_{st} az s és t csúcsok közötti legrövidebb utak száma, míg $\sigma_{st}(v)$ azon s és t közötti legrövidebb utak száma, amik áthaladnak v csúcson.

Közelség centralitás - Closeness Centrality: Egy csúcs közelség centralitása (röviden és a továbbiakban CC) számolható a csúcs és a gráfban szereplő többi csúcs közötti legrövidebb utak hosszának összegének reciprokaként [4]. A v csúcs normalizált CC értékét a következőképpen számítjuk ki:

$$CC(v) = \frac{N-1}{\sum_{u=1}^{N} d_G(v,u)}, \tag{1.3}$$

ahol $N = |V|$.

Harmonikus centralitás - Harmonic Centrality: A Közelség centralitáshoz hasonló mérték. Nem feltételen összefüggő gráfban a következő módon számítjuk:

$$HC(v) = \sum_{u=1}^{N} \frac{1}{d_G(v,u)}, \tag{1.4}$$

ahol $1/d(v,u) = 0$-nak van definiálva, ha v és u között nincs út. Normalizált verziójában $N-1$-gyel osztjuk az értéket [44].

PageRank: A web oldalak fontosságának mérésére fejlesztett centralitás. A szomszédsági mátrix néhány változtatásával kapott mátrix domináns sajátvektorának elemei lesznek a csúcsok értékei [53].

1.2.6. Egyéb fogalmak

Izomorfia: Azt mondjuk, hogy $G = (V, E)$ és $H = (V', E')$ gráfok izomorfak [8], ha létezik $f : V \to V'$ bijekció, hogy

$$(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E' \quad (\forall u, v \in V).$$

Monte Carlo-módszer: algoritmus, mely az ismételt véletlenszerű mintavételezéssel ad numerikus eredményt [46].

Lokális Optimumok Hálózata - Local Optima Network: Informálisan, a lokális optimalizálók pontjai (eredményei) a csúcsok, és az élek azon csúcsok között vannak definiálva, amelyeket egy kritikus pont (olyan stacionárius pont, ahol a Hesse mátrixnak pontosan egy negatív sajátértéke van [48]) választ el. Általános megfogalmazás található a [63]-ben. Az angol név rövidítéséből LON-nak nevezzük a későbbiekben.

1.2.7. Rövidítések

ds: Distance Sum, az adott csúcsba vezető legrövidebb utak hosszának összege, a CC érték reciproka.

K_n: n csúcsú teljes gráf, minden csúcs össze van kötve a gráf többi csúcsával.

P_n: n csúcsú útvonal gráf, az első és utolsó csúcs fokszáma 1, a többié 2.

S_n: $n+1$ csúcsú csillag gráf, $n-1$ darab csúcs fokszáma 1, egy csúcs fokszáma $n-1$.

WS: Watts-Strogatz algoritmussal készített gráf, kisvilág tulajdonsággal rendelkezik

BA: Barabási-Albert algoritmussal gyártott gráf, skálafüggetlen és fokszámsorozata a hatvány eloszlást (*power law distribution*) követi

CF: Cooper-Frize modellel készített gráf, skálafüggetlen tulajdonságú

2. fejezet

Gráf rekonstrukció

Ebben a fejezetben egy régóta vizsgált probléma eddig a szakirodalomban nem tanulmányozott verzióját szerettük volna a jelenlegi népszerű módszerektől eltérő módon megoldani a probléma jobb megértéséért. A feladatot specifikáltuk az egyszerűsítésért, melyből kiindulva általános esetek megoldását szeretnénk elérni.

A fejezetben ismertetett eredmények a [34] cikkben jelentek meg.

2.1. Bevezetés

A gráfelmélet egy hatékony eszköz, amelyet manapság gyakran használnak például adatok tárolására, bizonyos társas viselkedések modellezésére, események előrejelzésére, stb. A gráfok elegáns egyszerűsége, azaz hogy egy (egyszerű) gráf csúcsok és élek halmazaiból áll, vonza az érdeklődést az elméleti tanulmányok iránt csak úgy, mint az alkalmazásorientált kutatások felé. Általában az elmélet és az alkalmazás kölcsönösen szolgálja egymást motivációkkal. Ebben a fejezetben egy bizonyos típusú gráf rekonstrukciós problémával foglalkozunk, amely általában meglepően könnyen érthető. Ebben a felállásban korlátozott mennyiségű információt kapunk a gráf csúcsairól vagy éleiről, és a feladat egy olyan gráf megalkotása, amely csúcsai és élei teljesítik az adott tulajdonságokat. Erősen kapcsolódó téma konjektúra rekonstrukció, amelyet tanulmányoztak például a [36] és [61] cikkekben. A gráfok rekonstrukciója korlátozott mennyiségű információból, mint ahogy [54]-ban is megtalálható, érdekes, mert

– használható nagy méretű hálózatok tárolására tömörítve és

– használható generatív modellekként, azaz megadott tulajdonságú gráfok gyártására.

Az egyik legjellegzetesebb és legjobban kutatott probléma a témában a gráfok rekonstrukciója fokszámsorozatból, lásd [19, 38, 60, 67].

Egy gráf fokszámsorozatára gondolhatunk úgy, mint a gráf egy centralitási mértékére. Centralitás egy általános fogalom a hálózattudományban (alkalmazott gráfelmélet) [52]. A centralitási mérték egy olyan függvény, mely egy N csúccsal rendelkező gráfot használ bemenetként, és egy N hosszúságú vektort ad vissza, mely nemnegatív számokat tartalmaz. A vektorban lévő számok (mármint a centralitás értékek) jellemzik a gráf csúcsait, mivel lehetőséget adnak a csúcsok fontosságai szerinti rangsorolására. A fokszám centralitáson kívül a [15, 57] irodalmakban számos centralitási mértéket javasoltak.

Az ebben a fejezetben tárgyalt probléma most pontosítható. Adott egy nemnegatív, $\mathbf{c}$-vel jelölt N méretű vektor, amely eredménye egy G gráfon alkalmazott $C : G \to \mathbb{R}^N_{\geq 0}$ centralitási mértéknek. Keressünk G gráfot, amelyre $C(G) = \mathbf{c}$ teljesül.

Ahogy előzőleg említettük, az egyik első centralitási mérték, amelyet gráf rekonstrukcióra használtak a fokszámsorozat [20]. Egy kapcsolódó probléma a gráf rekonstruálhatóság, amiben azt kell eldönteni, hogy egy adott számsorozat egy gráf fokszámsorozat-e. Egy algoritmust bemutattak a [29] cikkben, később pedig a [26] cikkben. Utóbbi időben más hálózati tulajdonságok is lettek tesztelve a rekonstrukciós problémához, mégpedig a közöttiség centralitás (Betweenness Centrality) [11, 43], a spektrális értékek [9, 10, 35], és szomszédsági értékek [18].

Egy nagyon hasonló probléma, ha a bemeneti adat a 'belső' csúcsok[1] közötti távolság. A [27] cikkben egy algoritmust mutatnak be, amely megoldja a rekonstrukciós problémát súlyozott gráfokra (hurok mentesek) az összes 'külső' csúcspárok közötti távolságokból. A 'belső' csúcsok száma nem ismert, annyit adhatunk hozzá, amennyi szükséges. A [30] cikkben ugyanez, de specifikálták súlyozatlan fákra és azt tárgyalják, hogy mi a legkevesebb szükséges információ a probléma megoldásához.

Egy általánosabb módon a problémánkhoz kapcsolódik a mátrix rekonstrukció és a bináris rekonstrukció. A mátrix rekonstrukció során a feladat egy rejtett mátrix megtalálása valós értékű adatokkal a korlátozott számú ismeret elemekből [7]. Esetünkben a (szomszédsági) mátrixot kell rekonstruálni, amelynek bináris elemei vannak és egyik elem sem ismert. Bináris rekonstrukcióban a probléma az, hogy eltávolítsunk zajt a megfigyelt képből, ahol a zajok mennyisége ismert és időnként a zajos, megfigyelt mátrix sorai és oszlopai közötti kapcsolat is ismert [62, 64].

A fejezet további része a következőképpen van szervezve. A 2.2 alfejezetben bemutatjuk az összes definíciót és megadjuk a formális leírását a problémának az izomorf megoldások nehézségeinek rövid átbeszélésével. Ezek után a 2.3 alfejezetben néhány kapcsolódó szabályt formalizálunk és bizonyítunk be, melyek az algoritmusok megalkotásához kellettek. A három algoritmusunkat a 2.4 alfejezetben mutatjuk be. A pszeudokódokat és a részletes magyarázatokat néhány szemléltető példa támasztja alá. A 2.5 alfejezetben tárgyaljuk a kísérleti eredményeket. Végül a fejezetet összegezzük a 2.6 alfejezetben.

[1] A 'külső' csúcsok levelek, a 'belső' csúcsok a nem levelek.

2.2. Fogalmak

2.2.1. Közelség centralitás

Célszerű a csúcsok közelség centralitás értékeit CC vektorként hivatkozni, azaz mikor az értékeket egy N hosszú vektorba rendezzük és $CC[j]$-ként jelöljük.

Vegyük észre, hogy a normalizált CC követi az eredeti központisági fogalmat, nevezetesen azt, hogy a magasabb CC érték nagyobb fontosságot jelent. Az egyszerűség kedvéért azonban mi minden csúcsnál a legrövidebb utak összegét (ds - Distance Sum) vesszük csak figyelembe:

$$ds[j] = \sum_{k=1}^{N} d_G(j,k).$$

Továbbiakban a normalizás nélküli CC értékek reciprokával dolgozunk, melyek pozitív egész számok.

2.2.2. Probléma leírás

Adott egy pozitív egész számokból álló $\delta = (\delta_1, \ldots, \delta_N)$ vektor. A feladatunk, hogy találjunk egy $G = (V, E)$ fa gráfot $|V| = N$-nel, amelynél

$$\min \sum_{j=1}^{N} |ds[j] - \delta[j]|. \tag{2.1}$$

Egy másik népszerű célfüggvény a különbségek összegének négyzete. Problémafelvetésünkben a célfüggvény nem folytonos. Az abszolút különbség olyan információkat adhat nekünk, mint például, hogy hány lépés (él) hiányzik egyes csúcsokból. A fő célunk a nulla különbség elérése és nem a közelítése.

Fontos hangsúlyoznunk, hogy ebben a munkában többletinformációt használunk, vagyis feltételezzük, hogy ismerjük mely csúcsok levelek, azaz melyik csúcsoknak van pontosan egy szomszédjuk. Azonban nem ismerjük, kik ezek a szomszédok.

2.2.3. Nehézség az alkalmazásban

A CC vektor vagy annak reciprok vektorának kiszámítása egy tömörítésnek tekinthető: egy 2 dimenziós adatból (például szomszédsági mátrix vagy távolság mátrix) egy 1 dimenziós adatot (a távolság mátrix sorainak vagy oszlopainak összegei) készítünk. Ezen tömörítés alatt néhány adatot elvesztünk a szomszédsági mátrixban lévő felcserélhetőség miatt.

$$\begin{matrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{matrix}$$

2.1. ábra. *Három megegyező sor- és oszlopösszegű szomszédsági mátrix azonos pozíciójú részmátrixai.*

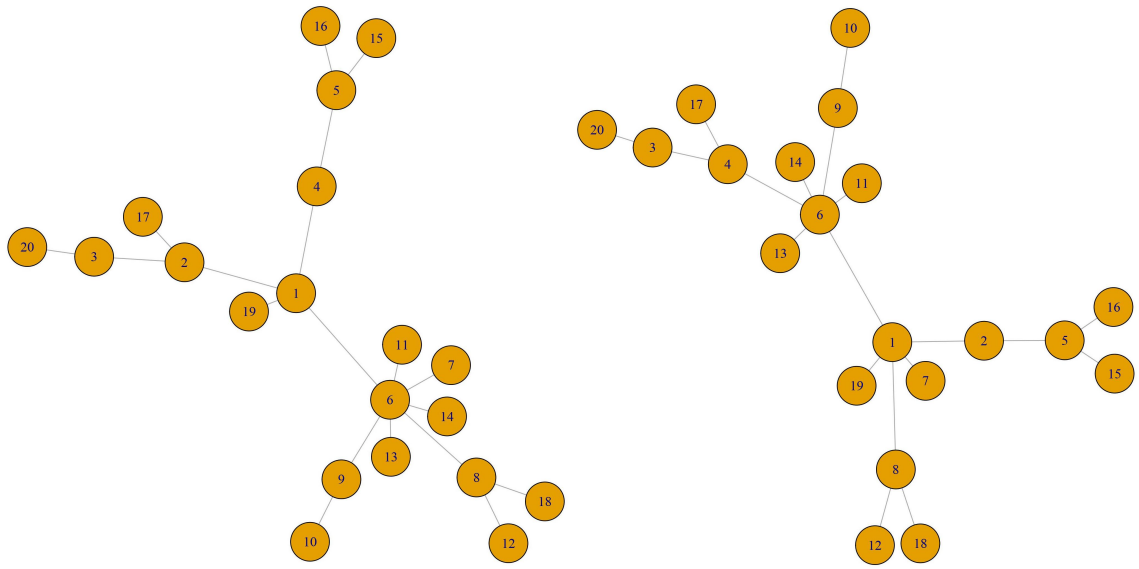
A 2.1. ábrán láthatunk példát arra, hogy azonos oszlop- és sorösszegek vannak a különböző gráfokhoz tartozó szomszédsági mátrixok részein. Így ha a mátrixok egyéb részei megegyeznek, akkor azonos CC értékeket adnak a különböző gráfok. Ha két szomszédsági mátrix azonos helyen lévő elemei eltérnek, hasonlóan mint ezek a részmátrixok, akkor újracímkézhetjük az egyik gráfot (megcserélhetjük a mátrix i-edik és j-edik oszlopait és sorait), hogy megkapjuk a másikat. Ezek izomorf megoldásai a rekonstrukció problémának.

Azon problémákban, ahol a csúcsok címkéi fontosak (például szociális hálókon való klaszterezés, melyben egy csúcs egy adott személyt jelent) nem tudunk egyetlen megbízható megoldást adni. A másik probléma a tömörítéssel, hogy a nem izomorf gráfok is tudnak megegyező CC vagy ds értékeket készíteni; egy szemléltető pédával szolgálunk az olvasónak a 2.2. ábrával. Tehát azokra a problémákra, amelyeket több izomorf megoldás is kielégít, nem tudunk egyértelmű eredményt adni. A fő probléma, hogy amíg le tudjuk gyártani az összes izomorf gráfot egy megoldásból az azonos értékű csúcsok címkéinek felcserélésével, addig a nem izomorf megoldásokért egy vagy több részét kell a gráfnak megváltoztatni még ismeretlen módon. Szintén megszámlálni a nem izomorf megoldásokat nem triviális. Ebben a munkában nem térünk ki a több megoldásos esetek kezelésére, egy elfogadható megoldás megtalálásával megelégszünk.

A 2.2. ábra nem izomorf fákat mutatunk, melyeknek a Közelségi központisági értékei megegyeznek. A 2.2. ábra (a) részén a legnagyobb fokszám 7, míg a 2.2. ábra (b) részén 6. Ha a fokszámsorozat különböző, akkor a gráfok nem lehetnek izomorfak.

2.3. Elméleti eredmények

Az első megfigyeléseink nyomán jutottunk a következő tételhez. A tétel teljesül az összes összefüggő, egyszerű gráfra, nem csak fákra.



(a) **(b)**

2.2. ábra. *Nem izomorf fák azonos közelség központisági értékekkel.*

2.1. Tétel. *Adott $G = (V, E)$ gráf és $(v_1, v_2) \in E$ él. Felhasználva a*

$$V_G(v_i, v_j) := \{v : v \in V,\ d_G(v, v_i) \leq d_G(v, v_j)\} \tag{2.2}$$

jelölést teljesül, hogy

$$ds[v_1] - ds[v_2] = |V_G(v_2, v_1)| - |V_G(v_1, v_2)|. \tag{2.3}$$

Bizonyítás. Először a (2.3) formulát bizonyítjuk fákra. Feltételezzük, hogy $(v_1, v_2) \in \in E$. Mivel G egy fa, így $d_G(v, v_1) \neq d_G(v, v_2)$ igaz minden $v \in V$-re. Ha $v \in \in V_G(v_1, v_2)$, akkor $d_G(v, v_2) = d_G(v, v_1) + 1$. Ráadásul ha $v \in V_G(v_2, v_1)$, akkor $d_G(v, v_1) = d_G(v, v_2) + 1$. Minden v csúcs $ds[v_1]$-hez és $ds[v_2]$-höz is hozzájárul ± 1 értékkel. Az előjelet, $+$ vagy $-$, a $d_G(v, v_1) < d_G(v, v_2)$ vagy $d_G(v, v_1) > d_G(v, v_2)$ határozza meg. A $ds[v_1] - ds[v_2]$ értéke függ a $|V_G(v_1, v_2)|$-től és $|V_G(v_2, v_1)|$-től. A $V_G(v_1, v_2)$-beli csúcsok $+1$-gyel járulnak hozzá a $ds[v_2]$-höz, mint $ds[v_1]$-hez, és $V_G(v_2, v_1)$-beli csúcsok -1-gyel járulnak hozzá $ds[v_2]$-höz, $ds[v_1]$-hez. Ha $E = \{(v_1, v_2)\}$, vagyis a gráfunknak csak egy éle van, akkor $ds[v_2] = ds[v_1]$. Ha más éleket és csúcsokat adunk a gráfhoz, akkor $ds[v_1]$ és $ds[v_2]$ úgy módosul, ahogy az előző mondatokban említettük. A $ds[v_1] + |V_G(v_1, v_2)| - |V_G(v_2, v_1)| = ds[v_2]$-ből következik (2.3).

Most bizonyítjuk (2.3)-t a nem-fa gráfokra. Ha $v \in V_G(v_1, v_2) \cap V_G(v_2, v_1)$, akkor v csúcs $ds[v_1]$-et és $ds[v_2]$-t is ugyanazzal az értékkel növeli, amely $|V_G(v_1, v_2)|$-ből és $|V_G(v_2, v_1)|$-ből számolható. A v csúcs $V_G(v_1, v_2)$ és $V_G(v_2, v_1)$ halmazokat 1-gyel, $ds[v_1]$

és $ds[v_2]$ értékeit ugyanazzal a számmal növeli, így (2.3)-beli egyenlőség mindkét oldalán a különbség nem fog változni. □

A 2.1. Tétel specifikálható fa gráfokra.

1. Lemma. *Ha $G = (V, E)$ egy fa, akkor*

$$ds[v_2] = ds[v_1] - N + 2 \cdot |V_G(v_1, v_2)| \tag{2.4}$$

minden $v_1, v_2 \in V$-re, ahol a V_G jelölés a (2.2)*-ban van definiálva.*

Bizonyítás. Tételezzük fel, hogy G egy fa és $|V| = N$. Ekkor $|V_G(v_1, v_2)|+|V_G(v_2, v_1)| = = N$. Tudjuk (2.3)-ből, hogy

$$ds[v_2] = ds[v_1] + |V_G(v_1, v_2)| - |V_G(v_2, v_1)|,$$

ennélfogva

$$ds[v_2] = ds[v_1] + |V_G(v_1, v_2)| - N + |V_G(v_1, v_2)|.$$

□

Egy csúcs ds értékét kiszámolhatjuk egy szomszéd értékéből és azon csúcsok halmazának a méretéből, amelyek ehhez a szomszédhoz vannak közelebb. Ezekből az észrevételekből, ha v_1 egy levél, akkor az egyetlen szomszédjának, v_2-nek, a ds értékére teljesülni kell, hogy $ds[v_2] = ds[v_1] - N + 2$ (itt használhatjuk azt a tényt, hogy $|V_G(v_1, v_2)| = 1$). A levelekhez pontosan kiszámolhatjuk a szomszédaik ds értékét, és meghatározhatjuk a legkisebb lehetséges szomszédok halmazát (izomorfiáig) egyszerű kivonással, mert ismerjük a $V_G(v_1, v_2)$ és $V_G(v_2, v_1)$ halmazok méreteit.

Nem-fa gráfokra is igaz a (2.3) egyenlet, azonban $|V_G(v_1, v_2)| + |V_G(v_2, v_1)| \neq = N$, így nem tudjuk kiszámolni egy szomszéd ds értékét a $|V_G(v_1, v_2)|$-ből (vagy $|V_G(v_2, v_1)|$-ből), mert nincs információnk hány csúcs helyezkedik el azonos távra az adott csúcstól és a szomszédjától.

A (2.4) egyenletből láthatjuk, hogy $ds[v_1] - ds[v_2]$ akkor éri el a maximumát, ha $|V_G(v_1, v_2)| = 1$ (nem lehet nulla vagy negatív). Így a legnagyobb szomszédos csúcsok közötti ds érték eltérés $N - 2$, és ebben az esetben az egyik csúcsnak levélnek kell lennie.

2.3.1. Levél csúcsok detektálása

Levél csúcsok sok információt adnak nekünk. Ha megtaláljuk az egyetlen szomszédját egy levélnek, akkor rengeteg potenciális élet dobhatunk el. Az input vektorból néhány csúcsról könnyedén el tudjuk dönteni hogy levél.

2. Lemma. *Ha $G = (V, E)$ egy fa, akkor mindazon $v \in V$ csúcsra, amelyekre teljesül, hogy*

$$\max_i(\delta_i) - (N-3) \leq ds[v] \leq \max_i(\delta_i) \tag{2.5}$$

leveleknek kell lenniük.

Bizonyítás. Tegyük fel, hogy $v_1, \ldots, v_k$ olyan csúcsok, amelyek teljesítik (2.5) egyenlőtlenségeket és

$$\max(\delta) = ds[v_1] \geq ds[v_2] \geq \ldots \geq ds[v_k].$$

Tekintsük a következő eseteket.

– A v_1 csúcsnak levélnek kell lennie (a középpontól távolabb nincs csúcs és G egy fa), illetve 2.1 tételből következik, hogy kapcsolódnia kell egy u csúcshoz, amelyre teljesül a $ds[v_1] - ds[u] = N - 2$. $u \notin \{v_1, \ldots, v_k\}$, kivéve ha $|V| = 2$.

– Minden $\ell = 2, \ldots, k$-ra v_ℓ csúcs levél kell legyen, mert nincs olyan nagyobb ds értékű csúcs, amelyhez kapcsolódhatna (az összes nagyobb értékű csúcs levél és egy olyan csúcshoz kapcsolódik, amelyik teljesíti a (2.4) egyenletet). A 2.1 tételt követve egy u csúcshoz kapcsolódik, amelyre $ds[v_\ell] - ds[u] = N - 2$.

– Ha u csúcs nem teljesíti a (2.5) egyenlőséget, akkor u lehet nem-levél. Lehetnek $w_1, w_2 \in V$ csúcsok, úgy hogy $0 \leq ds[w_1] - ds[u] \leq N - 2$ és $ds[w_1] \geq ds[u] \geq$ $\geq ds[w_2]$. Így w_1 egy levél vagy létezik út egy levéltől w_1-ig, és u a w_1 egy szomszédja. Az u csúcsnak kapcsolódnia kell w_2-höz (több mint egy szomszédja kell legyen, nem egy levél) vagy $ds[u] = \min_i(\delta_i)$. Ha $ds[u] = \min_i(\delta_i)$, akkor vagy u nem levél, mert legalább egy csúcs nagyobb értéket ad w_1-nek, mint u-nak, így ez a csúcs közelebb van u-hoz (létezik út, amely nem tartalmazza w_1-et, létezik más szomszédja), vagy $ds[w_1] = ds[u]$, ekkor $|V| = 2$ ($u = w_2$ és u teljesíti (2.5)-t).

Azok a csúcsok, amelyekre $ds < \min_i(\delta_i) + N - 2$ teljesül nem lehetnek levelek. Az 2.1. tételt használva v_1 és v_2 csúcsok nem lehetnek szomszédok, ha v_1 egy levél és $ds[v_1] - ds[v_2] \neq N - 2$. Így u csúcs nem lehet levél, ha $ds[u] < \min_i(\delta_i) + N - 2$, nincs olyan w csúcs u-val, ami teljesíteni tudná (2.3)-t, ha azt mondjuk, hogy $|V_G(u, w)| =$ $= 1$. □

2.4. Algoritmusok

Ebben az alfejezetben három algoritmust mutatunk be a közelség centralitás vektorból fa rekonstrukciójának problémájára. A 2.1 és 2.2 táblázatok tartalmazzák az összes függvényt, amelyeket a következő algoritmusok használnak. Az alábbiakban a javasolt módszerek leírása található a pszeudókódok megfelelő sorainak hivatkozásával.

2.1. táblázat. *Az algoritmusok leírásában használt függvények - 1. rész*

Függvény	**Input**	**Leírás**
Assignments	halmaz, szám	érték szerint kiválaszt elem(ek)et
BinPack	szám, szám	Ládapakolás Probléma megoldó
CheckPossibleNeighbours	lista	ellenőrzi, hogy alista egyik eleme sem üres
ClearAssignment	lista, listák listája	ellenőri, hogy a két paraméter elemszáma megegyezik-e
Combination	halmazok halmaza vagy halmazok listája	elkészíti az összes kombinációját az elemeknek, minden halmazból pontosan egy elemet használ
DivideByFirstValue	halmaz	a halmazt felosztja részhalmazokra az elem első értéke szerint
DsGen	gráf	egy gráf ds vektorát készíti el
DSChange	gráf, lista, szám, szám, lista	a csúcsok új ds értékét adja meg egy él újrakötése után
Edge	szám, szám	él objektumot hoz létre
	szám, lista	az első paraméterrel és egyessével a lista elemeivel készít él objektumokat
	lista, lista	él objektumokat hoz létre, mindkét lista i-edik eleme alkot egy párt

2.4.1. Maximum Presolver

Az első algoritmusunk célja, hogy egy közelítő megoldást találjon rövid idő alatt. Az ötlet azon a tényen alapul, hogy az 2.1. Tétel igaz a levelekre és szomszédjaikra. Egy (u, v) élre, ha sem u, sem v nem levél, akkor $ds[u] - ds[v] < N - 2$ igaz. Ennek az algoritmusnak a fejlesztésében azt a könnyen bizonyítható tényt használtuk, hogy szomszédos csúcsok ds értékének különbsége akkor a legnagyobb, ha az egyik csúcs levél. Ez a (2.4) egyenlet következménye.

A Maximum Presolver a 1. Algoritmusban van felvázolva, és az alfejezet további részében ennek soraira hivatkozunk.

Először rendezzük a ds vektor elemeit nem-növekvő sorrendbe. Legyen $vsorted$ a permutációs vektor, ami tartalmazza az indexeit az értékei szerint rendezett δ input vektornak (2. sor). Ezt a sorrendet használva készítünk egy $N \times N$-es táblázatot az összes ds érték egymástól vett különbségének tárolására. Ez a táblázat legyen D-vel jelölve (4. sor), így

$$D[i, j] = ds[i] - ds[j].$$

Tudjuk, hogy ha $(i, j) \in E$, akkor $ds[i] - ds[j] \leq N - 2$. Ezáltal a D táblázatban azok az értékek, amelyek nagyobbak mint $N - 2$, a diagonális elemek és a levelek oszlopaiban elhelyezkedők kicserélhetőek -1-re (ahogy a 5–8 sorokban szerepel). Ezek az elemek a nem lehetséges éleket jelölik. A táblázatban a $D[i, j] \geq 0$ azt jelöli, hogy a $(vsorted[i], vsorted[j])$ pár egy lehetséges él. Minden sor egy csúcsot jelöl, amit egy olyan oszlop által jelölt csúcshoz akarunk kötni, aminek a ds értéke kisebb vagy egyenlő. A táblázat utolsó sorát törölhetjük, mert az egyetlen oszlop, aminek nemnegatív eleme van (mielőtt helyettesítettünk volna -1-gyel) ebben a sorban, az az utolsó, azaz a diagonális elem. Egy fának $N - 1$ éle van, így iteratívan $N - 1$ lépést kell tennünk egy gráf készítésekor, ha egy lépésen belül két csúcsot kötünk össze.

1. Példa. *A szomszédsági mátrixból készített módosított D táblázat előállítására láthatunk példát a 2.3 ábrán. Megjegyezzük, hogy a módosítást követően egy $(N - 1) \times N$ méretű táblázatunk van.*

	1	2	3	4	5	6	7
1	0	1	2	1	1	2	1
2	1	0	1	2	2	1	2
3	2	1	0	3	3	2	3
4	1	2	3	0	2	3	2
5	1	2	3	2	0	3	2
6	2	1	2	3	3	0	3
7	1	2	3	2	2	3	0
CC	8	9	14	13	13	14	13

(a)

	14	14	13	13	13	9	8
14	0	1	1	1	1	5	6
14	0	0	1	1	1	5	6
13	-1	-1	0	0	0	4	5
13	-1	-1	0	0	0	4	5
13	-1	-1	0	0	0	4	5
9	-5	-5	-4	-4	-4	0	1
8	-6	-6	-5	-5	-5	-1	0

(b)

	14	14	13	13	13	9	8
14	-1	1	1	1	1	5	-1
14	-1	-1	1	1	1	5	-1
13	-1	-1	-1	0	0	4	5
13	-1	-1	-1	-1	0	4	5
13	-1	-1	-1	-1	-1	4	5
9	-5	-5	-4	-4	-4	-1	1

(c)

2.3. ábra. *(a) Egy gráf távolság mátrixa. (b) A rendezett ds értékek különbség táblázata. (c) Módosított különbség táblázat, alkalmatlan elemek -1-re cserélve, utolsó sor törölve*

Azt tapasztaltuk, hogy egy gráf éleinek többsége a módosított D sorainak maximum helyein helyezkedik el. Így felülről lefelé az összes sorban a maximum értékeket választjuk ki (9. sor). A levelektől a középpont felé építjük a gráfot. A 12–33. sorokban egy N csúcsú üres gráfhoz adunk éleket, amelyeket a végleges D sorainak és oszlopainak kiválasztásával teszünk meg.

```
1  Function Maximum Presolver(δ, leaves)
2      DSSort, vsorted := Sort(δ, "decrease", "index.return")
3      LeavesPos := Where(leaves == vsorted)
4      D := Outer(DSSort, DSSort, "-")      // like outer product but subtraction
5      D[D > (N-2)] := -1                                  // delete great values
6      D[i, i] := -1                                                 // no loops
7      D := D[1..(N-1), *]                                      // delete last row
8      D[*, LeavesPos] := -1                  // leaves have no bigger neighbour
9      MaxEdges := MaxInRow(D)        // maximum places in each row, list of
        lists
10     N := Length(δ)                                           // number of nodes
11     G := EmptyGraph(N)
12     for r in 1..(N-1) do                                     // step row by row
13         if Length(MaxEdge[r]) == 1 then
14             G := G + Edge(r, MaxEdge[r][1])
15         else
16             rowsame:=0                                  // same value row counter
17             while DSSort[r] == DSSort[r+rowsame] do
18                 rowsame := rowsame + 1
19             if rowsame > Length(MaxEdge[r]) then
20                 Δ := ∅                                  // values for balancing
21                 for j in 0..(rowsame-1) do
22                     Δ[j] := DsGen(G)[vsorted[r + j]] + Size(G, vsorted[r+j])
23                 bGroups := BinPack(Δ, Sum(Δ) / Length(Δ))  // items, capacity
                    of a bin
24                 for j in 1..Length(bGroups) do
25                     rnodes := vsorted[r + bGroups[j]]        // rows for a column
26                     G := G + Edge(rnodes, vsorted[MaxEdge[r] + j])
27             else                                                // #row ≤ #column
28                 cols := Sample(1..Length(MaxEdge[r])
29                 for i in 0..(rowsame-1) do
30                     rnode := vsorted[r + i]                    // node of a row
31                     cnode := vsorted[cols[MaxEdge[r] + i]]   // node of a column
32                     G := G + Edge(rnode, cnode)
33             r := r + rowsame                                    // instead of r+1
34     return G, D, vsorted
```

1. Algoritmus: Maximum Presolver algoritmus

2.2. táblázat. *Az algoritmusok leírásában használt függvények - 2. rész*

Függvény	**Input**	**Leírás**
EmptyGraph	szám	létrehoz egy gráfot élek nélkül
EstimateNeighbourDS	gráf, lists	megadja a ds értékeit a listabeli elemeknek, a 2.4 egyenletet használja
IntegerSplitUps	szám, szám	megadja az összes megoldását a speciális Change-making problémának, ahol az érmék számának minimalizálása nem cél
Length	lista vagy halmaz	megadja az elemek számát, ha az input összetett, akkor az első szint elemszámát
MaxInRow	mátrix vagy táblázat	ad egy listák listáját, amely tartalmazza a soronkénti maximum helyeket
NodesFromAssignment	halmaz, szám, szám	megadja az adott elhelyezésből az adott csúcs szomszédait
Outer	lista, lista, operátor	mátrixot vagy táblázatot készít
Sample	lista	véletlenszerűen rendezett listát ad
Sort	lista, string	visszaadja a listát az adott rendezés szerint
	lista, string, string	visszaadja a listát és a permutációs vektort az adott rendezés szerint
Sum	lista	megadja a lista elemeinek összegét
Unique	lista vagy halmaz	visszaadja a lsitát vagy halmazt értékek ismétlése nélkül
Where	lista	megadja az indexeket, ahol a lista igaz elemeket tartalmaz
Where.Max, Where.Min	lista	megadja az első maximum illetve minimum érték indexét

Ha több mint egy maximum van (15. sor), a Maximum Presolver algoritmus ebben az r_1 sorban megjelöli $c_1, c_2, \ldots, c_m$-mel azokat az oszlopokat, amelyeknek ugyanezen maximum értékük van. Majd elkezdi vizsgálni a lejjebb elhelyezkedő sorokat (16. sor): kiválasztja azokat a sorokat ($r_2, r_3, \ldots, r_n$), amelyeknek $c_1, c_2, \ldots, c_m$ oszlopaiban ugyanazok a maximum értékek szerepelnek, mint r_1 sorban. A sorrendbe rendezés miatt, ha egy sor nem felel meg, akkor ettől lentebb lévők sem fognak.

A 22. sorban kiszámolja az összes $vsorted[r_j]$ ($j \in \{1, 2, \ldots, n\}$) csúcs aktuális ds értékét és a G_j részgráfok méretét, amiket eddig épített. A $G_j = (V_j, E_j)$ azon összefüggő gráfokat jelöli, amelyek tartalmazzák a $vsorted[r_j]$ csúcsot. Ezen részgráfok ds vektora ds_j-ként van jelölve. Ezekkel a számokkal becsüljük egy $\{vsorted[c_k] \ : \ k \in$

$\in \{1, 2, \ldots, m\}\}$ csúcs ds értékének $\Delta_j = ds_j[vsorted[r_j]] + |V_j|$ növekedését, ha egy $vsorted[r_j]$ csúcsot kötnénk hozzá. Általában az épülő gráf nem összefüggő, ezért használjuk a részgráfok méreteit, és a ds vektor csak az utakon keresztül elérhető csúcsokból számolódik.

Mostmár minden r_j sorhoz van Δ_j értékünk. Felosztjuk a sorokat az oszlopok között azon szabály szerint, amely azt mondja, hogy mindegyik oszlophoz rendelt Δ_j-k összege lehető legjobban közelítse az egyenlőt (23. sor). Kiegyensúlyozzuk a $vsorted[c_k]$ csúcsok aktuális ds értékét (lépés előtt 0, mert nem volt szomszédjuk), amelyek értéke megegyezik a δ input vektorban. Ezt a kiegyensúlyozási problémát Ládapakolási problémaként [45] oldjuk meg, ahol a ládák mérete megegyezik az értékek összege osztva az értékek számával. Ha az érték nagyobb, mint egy láda mérete, akkor ez az érték megtelített egy dobozt, és több érték behelyezését ebbe a dobozba nem engedünk meg.

2. Példa. *Egy szemléltető példa látható a 2.4. ábrán. Két blokkunk van, ahol a kiegyensúlyozást használtuk:*

– *A* 9*,* 16 *nevű sorok és az* 5*,* 6 *nevű oszlopok alkotják az első blokkot. Mindkét*

ID			19	9	16	11	8	12	14	18	15	5	6	17	20	13	2	7	10	4	3	1
	ds		83	79	79	69	63	63	63	63	65	61	61	67	67	53	51	49	45	45	49	35
		Input	88	84	84	74	72	72	72	72	70	66	66	62	62	58	56	54	54	50	44	40
19	83	88	-1	-1	-1	-1	-1	-1	-1	-1	18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	79	84	-1	-1	-1	-1	-1	-1	-1	-1	14	18	18	-1	-1	-1	-1	-1	-1	-1	-1	-1
16	79	84	-1	-1	-1	-1	-1	-1	-1	-1	14	18	18	-1	-1	-1	-1	-1	-1	-1	-1	-1
11	69	74	-1	-1	-1	-1	-1	-1	-1	-1	4	8	8	-1	-1	-1	18	-1	-1	-1	-1	-1
8	63	72	-1	-1	-1	-1	-1	-1	-1	-1	2	6	6	-1	-1	-1	16	18	18	-1	-1	-1
12	63	72	-1	-1	-1	-1	-1	-1	-1	-1	2	6	6	-1	-1	-1	16	18	18	-1	-1	-1
14	63	72	-1	-1	-1	-1	-1	-1	-1	-1	2	6	6	-1	-1	-1	16	18	18	-1	-1	-1
18	63	72	-1	-1	-1	-1	-1	-1	-1	-1	2	6	6	-1	-1	-1	16	18	18	-1	-1	-1
15	65	70	-1	-1	-1	-1	-1	-1	-1	-1	-1	4	4	-1	-1	-1	14	16	16	-1	-1	-1
5	61	66	-1	-1	-1	-1	-1	-1	-1	-1	-4	-1	0	-1	-1	-1	10	12	12	16	-1	-1
6	61	66	-1	-1	-1	-1	-1	-1	-1	-1	-4	0	-1	-1	-1	-1	10	12	12	16	-1	-1
17	67	62	-1	-1	-1	-1	-1	-1	-1	-1	-8	-4	-4	-1	-1	-1	6	8	8	12	18	-1
20	67	62	-1	-1	-1	-1	-1	-1	-1	-1	-8	-4	-4	-1	-1	-1	6	8	8	12	18	-1
13	53	58	-1	-1	-1	-1	-1	-1	-1	-1	-12	-8	-8	-1	-1	-1	2	4	4	8	14	18
2	51	56	-1	-1	-1	-1	-1	-1	-1	-1	-14	-10	-10	-1	-1	-1	-1	2	2	6	12	16
7	49	54	-1	-1	-1	-1	-1	-1	-1	-1	-16	-12	-12	-1	-1	-1	-2	-1	0	4	10	14
10	45	54	-1	-1	-1	-1	-1	-1	-1	-1	-16	-12	-12	-1	-1	-1	-2	0	-1	4	10	14
4	45	50	-1	-1	-1	-1	-1	-1	-1	-1	-20	-16	-16	-1	-1	-1	-6	-4	-4	-1	6	10
3	49	44	-1	-1	-1	-1	-1	-1	-1	-1	-26	-22	-22	-1	-1	-1	-12	-10	-10	-6	-1	4

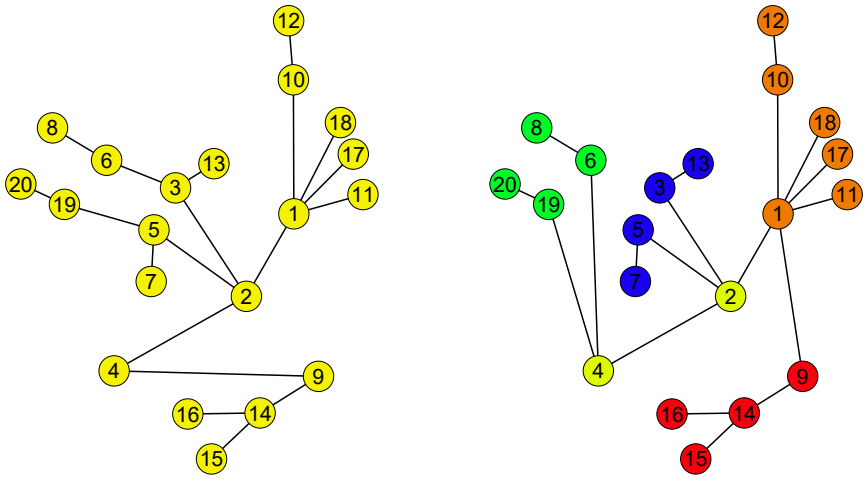
2.4. ábra. *Végső módosított különbség táblázat, a kiemelt értékek lettek a Maximum Presolver algoritmus által az élek meghatározásához*

sornak ugyanaz az értéke, ami 1. Nincsenek szomszédaik, izolált csúcsok. Mindkét oszlop egy-egy sort kap. Ez volt a 27. sor esete.

– *A második blokkot a* 8, 15 *nevű sorok és a* 7, 10 *nevű oszlopok határolják. A blokk első négy sora a* $\Delta_j = 0 + 1$ $(j = 1{,}2{,}3{,}4)$ *értékeket, az utolsó* $\Delta_5 = ds_5[15] + + 2 = 3$ *értéket kapja, mert a* 15*-ös csúcsnak van egy szomszédja (*11*-es csúcs) és a részgráfjának,* G_5*-nek, a mérete 2. Ezáltal a* 7*-es sor értéke 3, a* 10*-es soré 4, és ennél jobban nem egyensúlyozható ez az eset. Ez volt a 19. sor esete.*

2.4.2. Corrector

Ha a Presolver algoritmus készít egy G_P gráfot, amely a (2.1) célfüggvényünkön egy el nem fogadható alacsony értéket ad, akkor használhatunk egy javító algoritmust a G_P gráfon a megoldás minőségének javításáért. Jellemzően néhány csúcs elfogadhatóan kapcsolódik, az elvárt ds értéket adják. A többi csúcs csoportosulhat összefüggő részgráfokban, amelyekben a csúcsok ds értéke azonos módon tér el az input vektorbeli elemtől. Ha újrakötünk egy teljes részgráfot egy másik csúcshoz G_P-ben, akkor a részgráf összes csúcsának ds értéke azonos módon változik. Az összes többi csúcstól vett távolságok ugyanúgy változnak. Egy illusztratív példa szerepel a 2.5 ábrán.



2.5. ábra. *A rekonstruálandó eredeti fa gráf (bal oldalt) és a Maximum Presolver eredménye, amelyen a csúcsok színe a kívánt eredménytől való eltérést mutatja.*

A Corrector leírása a 2. Algoritmusban található, és a következőkben ennek sorait hivatkozunk.

A heurisztikus Corrector az aktuális ds értékek és a δ input vektor elemenkénti eltérése szerint választ csúcsot, azt amelyik a legnagyobb eltérést okozza. Legyen ez a kiválasztott csúcs y (8. sor). Kiválasztjuk azt a szomszédját, amelyiknek a legkisebb a ds értéke, legyen az z_1 (10. sor). Az (y, z_1) él törlésével két fa gráfot kapunk (16. sor).

```
1  Function Corrector(G, δ, D, vsorted, imax)
2      i:= 0                                                  // iteration counter
3      while i < imax do
4          ds:= DsGen(G)
5          dsDiff:= δ - ds
6          if dsDiff == 0̄ then
7              break
8          y:= Where.Max(dsDiff)                                  // index in δ
9          neighOfY:= Neighbours(y)
10         z1:= neighOfY[Where.Min(δ[neighOfY])]
11         ySorted:= Where(y == vsorted)                  // index in DSSort
12         z1Sorted:= Where(z1 == vsorted)                // index in DSSort
13         z2Group:= Where(D[ySorted, *] > 0) \ { z1Sorted }
                                                          // select from 1 row
14         newDSDiff:=
             |δ[interestednodes] − DSChange(G, vsorted[z2Group], y, z1, ds)|
15         z2:= choose z2 node by rule
16         G:= G − Edge(y, z1)
17         G:= G + Edge(y, z2)
18         i := i + 1
19     return G
```

2. Algoritmus: Általános Corrector algoritmus

Az egyik fa tartalmazza y-t, a másik z_1-et. Ezen vágás után az algoritmus kiválaszt egy csúcsot abból a fából, amelyik z_1-et tartalmazza. Legyen ez a csúcs z_2. A z_2 csúcsot az y lehetséges szomszédjai közül választjuk ki, melyeket a D különbség táblázat ír le. Hozzuk létre az (y, z_2) élet (17. sor). Az eredmény egy fa gráf lesz, mivel két fa gráfot egyesítettünk egy éllel anélkül, hogy kört alkottunk volna.

A célunk, hogy csökkentsük az eltérést az aktuális ds értékek vektora és az input vektor között (lásd 6. sor), de ez egy vágással sokszor nem lehetséges.

Ezért néhány heurisztikát kell használni. A következő négy megközelítést vizsgáltuk:

1. A leghatékonyabb verzió, amelyet készítettünk és teszteltünk, a $\underline{Z}$-Corrector. Ebben a változatban a z_2 csúcsot az a szabály választja ki, amely szerint a lehetséges csúcsok közül (kivéve z_1) a legkisebb eltérést adja a z_2 új ds értéke és $\delta[z_2]$ között. A 2 algoritmus 14. sorában az $interestednodes$ a $vsorted[z2Group]$-ból származó csúcsok, és a $DSChange()$ függvény a $vsorted[z2Group]$-beli csúcsok új ds értékeként adja meg. A 15. sort lecserélnénk a

   ```
   z2:= vsorted[z2Group[Where.Min(newDSDiff)]]
   ```

   sorra.

2. Az $\underline{Y}$-Corrector algoritmus z_2-t úgy választja a lehetséges csúcsok (kivéve z_1) közül ki, hogy az y új ds értéke és a $\delta[y]$ között a lehető legkisebb legyen az eltérés. A 2 algoritmus 14. sorában az $interestednodes$ az y-ra vonatkozik, a $DSChange()$ algoritmus az y-nak a z_2 kiválasztása szerinti új ds értékeit adja. A 15. sort

```
z2 := vsorted[z2Group[Where.Min(newDSDiff)]]
```

sorra cserélnénk.

3. Az $\underline{R}$-Corrector algoritmusban a z_2 kiválasztása uniform véletlen módon történik a lehetséges csúcsok közül. A 2. Algoritmusban a 14. sor szükségtelen és a 15. sort módosítanánk a

```
z2 := vsorted[Sample(z2Group)[1]]
```

sorra.

4. A negyedik algoritmust $\underline{F}$-Corrector-nak neveztük el, amely úgy dönti el z_2 kiválasztását, hogy az új ds értékek és a δ vektor különbségének normáját minimalizálja. A 2. Algoritmus 14. sorát

```
                    for(x in z2Group){
newDSDiff[x] := ‖ δ - DSChange(G, x, y, z1, ds)‖}
```

-re és a 15. sorát

```
Z2 := vsorted[z2Group[Where.Min(newDSDiff)]]
```

-re változtatjuk.

3. Példa. *A 2.6. ábrán egy példát láthatunk a z_2 kiválasztására. A felső táblázat a D módosított különbség táblázat egy része. Láthatjuk a csúcsok sorszámát (ID), az aktuális ds értékeiket, a δ input vektorbeli értéküket és az input értékek különbségét (néhány $-$ -1-re lett cserélve). A sort (vagy y csúcsot) az aktuális és a kívánt ds értékek közötti legnagyobb különbség szerint lett kiválasztva (az első maximális érték, ha több lenne). Ez a 8-as csúcs. A kékkel kiemelt különbségek a jelenlegi választott sor-oszlop párok. A 8-as csúcsot a 10-essel párosítottuk. Így a 10-es csúcs a z_1, az a csúcs, amelyik kapcsolódik az elmozgatni kívánt részgráfhoz. Az összes lehetséges új kapcsolódó csúcsnak a sorban lévő értéke nemnegatív. A csúcsok, amelyek közül kerül ki z_2, az új csatlakozási pont. Ezek a 15-ös, 5-ös, 6-os, 2-es és 7-es csúcsok. Az alsó táblázatból láthatjuk az y és*

ID			15	5	6	17	20	13	2	7	10	4
	ds		65	61	61	67	67	53	51	49	45	45
		Input	70	66	66	62	62	58	56	54	54	50
11	69	74	4	8	8	-1	-1	-1	18	-1	-1	-1
8	63	72	2	6	6	-1	-1	-1	16	18	18	-1
12	63	72	2	6	6	-1	-1	-1	16	18	18	-1
14	63	72	2	6	6	-1	-1	-1	16	18	18	-1

ID	15	5	6	2	7
Y	80	76	76	67	65
Z2	62	58	58	49	47

2.6. ábra. *Az Y-Corrector és a Z-Corrector működésére példa*

z_2 új ds értékeit, ha az ID sorban szereplő csúcsot választottuk z_2-nek. A pirossal írt számok vannak legközelebb a megfelelő δ-beli értékekhez. Az y sorában minden érték a 8-as csúcs inputbeli értékéhez van hasonlítva. A z_2 sorában az értékek az oszlopok által meghatározott saját csúcsuk kívánt értékéhez. Az Y-Corrector az 5-ös csúcsot választja, míg a Z-Corrector a 2-eset a részgráf új csatlakozási pontjának a régi helyett (10-es csúcs).

Ahogy korábban említettük, a Maximum Presolver melegindításra (*warm start*) is használható. A Corrector helyett használhatunk egy népszerű metaheurisztikát (pl. genetikus algoritmus, multi-start algoritmus, szimulált hűtés, tabu search, ABC, PSO, ACO). Ezen algoritmusok némelyikének megvizsgálása az egyik jövőbeli munkánk.

2.4.3. Recursive Solver

Ezt az algoritmust a pontos megoldásra terveztük. Az eredmény lehet a G fa gráf, amelynél $\sum_{j=1}^{N}|ds[j]-\delta[j]|=0$, vagy a '`Not Solved`' szöveg. A 2.1. Tételt és a (2.4) egyenletet használjuk ebben az algoritmusban. Tudjuk, hogy mely csúcsok fokszáma 1. Mint az előző algoritmusunkban, a Maximum Presolver-ben (2.4.1 alfejezetben), fát építünk a levelektől a központig. Minden $l \in V, degree(l) = 1$-re tudjuk $ds[l]$-t és $V_G(l,v)$ $(v \in V, (l,v) \in E)$-t, még akkor is, ha nem ismerjük v-t. Ezekből az

értékekből számolhatjuk $ds[v]$-t (nem v-t). Több mint egy csúcs teljesítheti a feltételt, hogy v legyen, mert a ds értékük megegyezhet. Ebben az esetben választanunk kell egy csúcsot. Ha két csúcs ds értéke meg is egyezik attól a fokszámukra nem kell, hogy ez igaz legyen. Így nem minden megfelelő ds értékű csúcs vezet egy elfogadható megoldáshoz.

A Recursive Solver a 3. és 4. Algoritmusokban van leírva. A Recursive Solver-hez készített többi algoritmus az 5, a 6 és a 7.

2.4.3.1. Inicializáció

Az inicializálási rész a 3. Algoritmusban van leírva. Első lépésben készítünk egy üres N csúcsú G gráfot (3. sor). Ezt követően, ha egy levél csúcs egyértelműen egy (nem-levél) csúcshoz tartozik, azaz egyetlen csúcs teljesíti a (2.3) egyenletet, akkor összekötjük őket. Ezeket a használt leveleket a Λ halmazba bevesszük (4. - 11. sor). Majd a 12. sorban az algoritmus az $usds$-ként jelölt listába helyezi csökkenő sorrendben, duplikációk nélkül a δ-beli elemek közül azokat, amelyek a nem-levél és még fel nem használt levél csúcsokhoz tartoznak. Végül a 13. sorban meghívódik a rekurzív rész (a 4. Algoritmus) a G gráfra a *step* $= 1$ paraméterrel.

```
1  Function RecursiveSolver(δ, leaves)
2      N:= Length(δ)                                   // global variable
3      G:= EmptyGraph(N)                               // global variable
4      Λ:= ∅                        // global variable, set of used leaves
5      for leaf in leaves do
6          ds_neig:= EstimateNeighbourDS(G, leaf)      // use Theorem 2.1
7          neighbours:= Where(δ == ds_neig)            // possible neighbours
8          if Length(neighbours) == 1 then
9              G:= G + Edge(leaf, neighbours)
10         else
11             Λ:= Λ ∪ leaf
12     usds:= Unique(Sort(δ[{1..N} \ Λ], "decrease"))  // global variable
13     G:= Recursive(G, 1)
14     return G
```

3. Algoritmus: Recursive Solver inicializáló része

2.4.3.2. Rekurzív hívás

Az inicializálás után a 4. Algoritmust használjuk. A *step* változó jelöli, hanyadik $usds$ elemet használjuk (a rekurzió mélysége). A *step* $= 1$ értékkel indítjuk.

```
1  Function Recursive(G, step)
2      if step == Length(usds) then
3          if Length(Where(δ == usds[step])) == 2 then          // 2 roots in G
4              G:= G + Edge(usds[step])                  // connect the last 2 nodes
5              if DsGen(G) == δ then
6                  return G
7              else
8                  return False                                  // failed branch
9      nodes:= Where(δ == usds[step]) \ Λ
10     ds_neig:= EstimateNeighbourDS(G, nodes)
11     η:= ∅                                                    // list of sets
12     for x in 1..Length(ds_neig) do
13         η[x]:= Where(δ == ds_neig[x]) \ leaves
14     if !CheckPossibleNeighbours(η[x]) then                  // check emptiness
15         return (False)
16     if ClearAssignment(nodes, η) then
17         G:= Recursive(G + Edge(nodes, η), step + 1)
18         return G
19     else
20         dispositionSet:= GenerateDispositions(nodes, η, ds_neig)
21         for edges in dispositionSet do
22             G2:= G + edges
23             G2:= Recursive(G2, step + 1)
24             if G2 != False then
25                 return G2
26         return False
```

4. Algoritmus: Rekurzív rész

Egy rekurzív lépésben először ellenőrizzük, hogy az algoritmus eljutott-e az $usds$ utolsó eleméhez (2. sor). Ha igen, akkor meg kell vizsgálnunk, hogy az eredeti gráfnak van-e két olyan csúcsa, amelyeknek a ds értéke megegyezik és a legkisebbek a δ-ban. Ha ez igaz, akkor ezek a csúcsok a középpont és össze kell őket kötnünk (4. sor). Ha elértük az $usds$ utolsó elemét, akkor elkészítettük mind az $N-1$ élet, így azt kell ellenőriznünk, hogy az épített G gráf ds értékei és a δ megegyezik. Ha igen, akkor befejeződött az algoritmus. Ha nem, akkor vissza kell lépnünk a rekurzióban. A visszalépést a *False* értékkel oldottuk meg a pszeudókódban (5 - 8. sor).

Ha a $step$ nem az $usds$ végét jelöli, akkor összegyüjtjük azokat a csúcsokat, amelyek ds értéke $usds[step]$, kihagyva a használt leveleket (Λ halmazban szereplő csúcsok) (9. sorban a $nodes$ halmaz). A 2.1 tételt használjuk, hogy megbecsüljük az összegyüjtött csúcsok ($nodes$) szomszédjainak ds értékét az élekre vonatkozó jelenlegi feltételezéseinkkel (korábbi választások az algoritmus elágazásakor) (ds_{neig} a

10. sorban). Kiválasztjuk a lehetséges szomszédokat a becsült ds értékeket használva. Egy szomszéd nem lehet levél. A $|V_G(x,y)|$ érték eltérhet néhány x esetén, $x \in \in nodes, (x,y) \in E, ds[x] \geq ds[y]$, így $ds[y]$ is eltérhet néhány x-nél. Ezen okból készítünk egy halmazok listáját, η névvel (13. sor). Minden halmaz egy becsült ds értékhez tartozik (vagy egy $nodes$ beli csúcsok csoportjához, ahol egy csoporton belül a $|V_G(x,y)|$ érték megegyezik). Egy halmaz elemei, $\eta[x]$ elemei, az összes lehetséges csúcs (nem levél), amelyek ds értéke megegyezik az adott becsült értékkel, $ds_{neig}[x]$-szel. Ha egy halmaz üres, akkor egy vagy több $nodes$-ban szereplő csúcsnak nincs lehetséges szomszédja, így az aktuális állapotból nem tudunk készíteni egy elfogadható fát. Vissza kell lépnünk a rekurzióban és egy másik elágazást választani (14. és 15. sor).

Ha minden $\eta[x]$ halmaznak egyetlen eleme van, akkor a $nodes$-ban szereplő összes csúcshoz egyértelműen rendelhetünk egy szomszédot. Folytathatjuk a következő rekurzív lépéssel az algoritmust. Az eredmény elérhető lesz, mikor a mélyebb szinteken lévő lépések befejeződtek (16. - 18. sor). Ha néhány halmaznak több mint egy eleme van, akkor a csúcsok és a lehetséges szomszédaik közötti élek összes elhelyezését el kell készítenünk (20. sor). Az algoritmus ezen a ponton elágazik. Kipróbálunk egy elhelyezést és folytatjuk az rekurziót (22. és 23. sor). Ha egy elágazás elfogadható eredménnyel (azaz nem $False$-szal) tér vissza, akkor abbahagyjuk a megmaradt elágazások vizsgálatát. Ekkor az algoritmus elérte az $usds$ végét, és az épített gráf ds vektora megegyezik a δ vektorral. Az eredményt a rekurzió felsőbb szintjére küldjük (24. és 25.sor). Ha az összes készített elhelyezés sikertelen volt, akkor visszább lépünk a rekurzióban $False$ eredménnyel (26. sor). Egy felsőbb szinten kipróbálunk egy másik elágazást (az élek egy másik elhelyezését).

4. Példa. *Példát láthatunk a rekurzív hívás néhány lépésére az 2.7 - 2.14. ábrákon.*

2.4.3.3. Elhelyezések

A Recursive eljárásunk ezen része a 5. Algoritmusban van leírva. Az élek elhelyezéséhez szükségünk van a csúcsokra ($nodes$), amelyeket akarjuk kötni más csúcsokhoz a jelenlegi lépésünkben, a halmazok listájára (η), ami tartalmazza a lehetséges szomszédokat és a lehetséges szomszédok ds értékére (ds_{neig}). Egymástól függetlenül létrehozzuk mindegyik halmazhoz az összes lehetséges elhelyezést. Készítünk egy listát ($dispSet$), ami tartalmazza az érvényes elhelyezéseket. A lista elemei halmazok lesznek az η-hoz hasonlóan (2. sor). Legyen $nodes_i$ a $nodes$ azon részhalmaza, amely az η i-edik halmazához (η_i a 5. Algoritmusban) tartozik. Legyen ns és ms a $nodes_i$ és η_i elemszáma (3 - 7. sor).

5. Példa. *A 2.16. ábrán láthatunk egy eredeti gráfot (nem az algoritmusunk által gyártott), amelynek a csúcsai a ds értékekkel van címkézve. A példa azt mutatja meg, hogy*

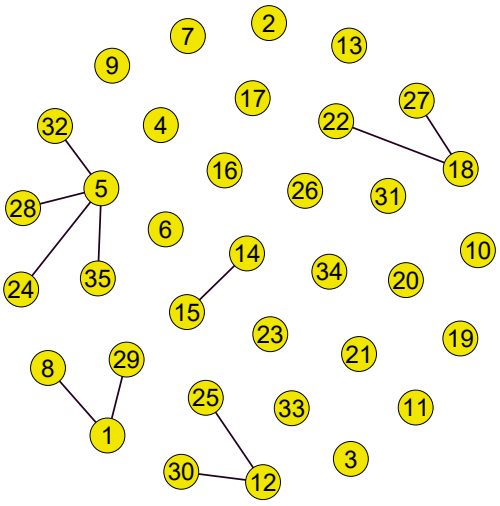
a Recursive algoritmus több részhalmazt (η_i) készítene az $usds$-ben lévő egy vizsgált ds értékhez, mert a részgráfok méretei (vagy $V_G(v_1, v_2)$ értékek) különböznek.

Most ns darab csúcsot kell ms darab csúcshoz rendelnünk. Szükségünk van az összes nem izomorf hozzárendeléshez. Láthatunk egy példát a 2.15. ábrán. Egy ekvivalens probléma, hogy ns-t szétosztjuk ms darab nemnegatív egész számra. Az összes lehetséges permutáció nélküli szétosztásra szükségünk van, az elemek ismétlődhetnek[2] (8. sor).

Minden elhelyezés növeli az algoritmusban az elágazások számát. Alkottunk két vágó metódust (a 2.4.3.4. alfejezetben vannak tárgyalva) azért, hogy csökkentsük az elágazásokat. Ezek a metódusok a készített elhelyezéseket megszűrik (9. sor). A megmaradt elhelyezéseket a $dispSet[i]$-be helyezzük (10 - 12. sor). Az η minden i-edik halmazához van egy elhelyezés halmazunk, $dispSet[i]$. Az összes halmazból származó egy-egy elemek kombinációja adja a $nodes$ egy teljes elhelyezését (13. sor).

2.4.3.4. Vágások

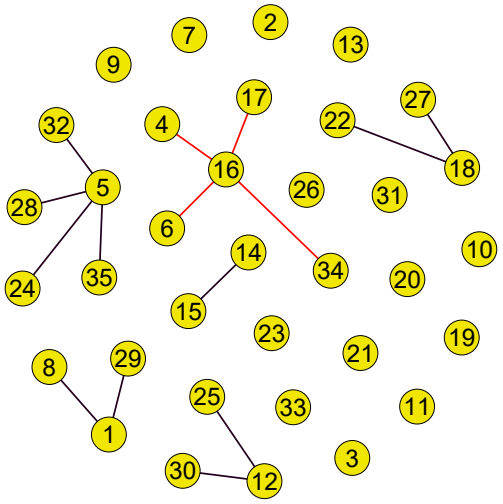
Két vágó (prune) metódust teszteltünk. Mindkettő a generált egész sorozatokat (σ a 5 algoritmus 8. sorában) és az adott halmaz első csúcsát használja. A vágó algorit-



usds	csúcsok
151	4 6 17 34
149	14 19 21 23 26 31 33
141	18
118	3 9 16 24 28 32 35
116	7 10 11 20
114	12
112	13
89	1
86	2
85	5

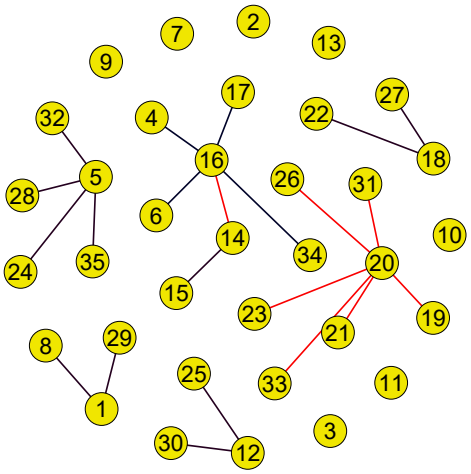
2.7. ábra. *A Recursive eredménye az inicializálás után.*

[2]Ez egy change-making-szerű probléma [66]



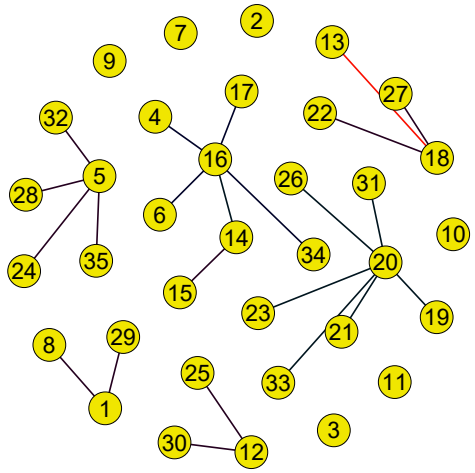
	usds	csúcsok
▶	151	4 6 17 34
	149	14 19 21 23 26 31 33
	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.8. ábra. *A Recursive eredménye az 1. lépés után; 1. elágazás a 4-ből az 1. szinten.*



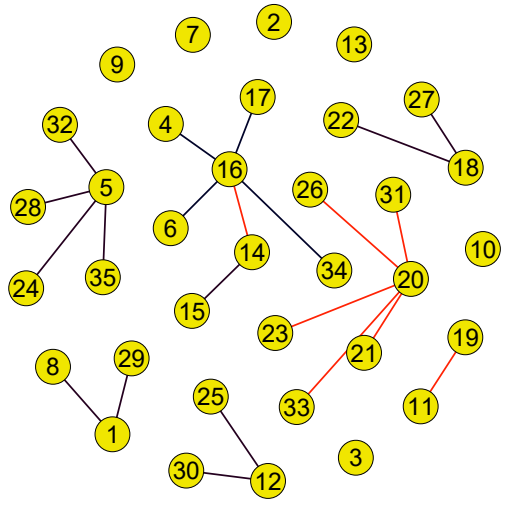
	usds	csúcsok
	151	4 6 17 34
▶	149	14 19 21 23 26 31 33
	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.9. ábra. *A Recursive eredménye a 2. lépés után; 1. elágazás a 18-ból a 2. szinten.*



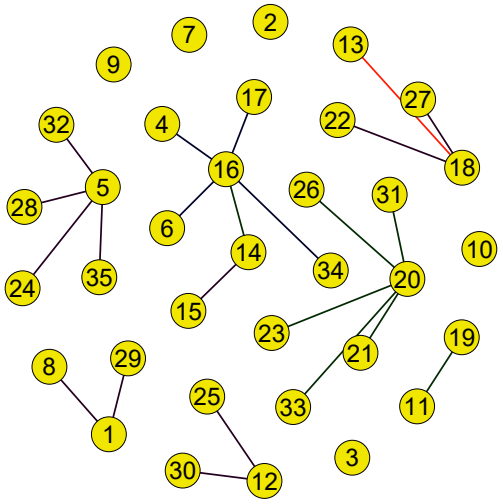
	usds	csúcsok
	151	4 6 17 34
	149	14 19 21 23 26 31 33
▶	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.10. ábra. *Eredmény a 3. lépés után. Nincs elágazás, a 4. lépésnél zsákutcába jut. Nincs megfelelő ds értékű lehetséges szomszéd a δ-ban egy vagy több csúcsnak a 4. szinten.*



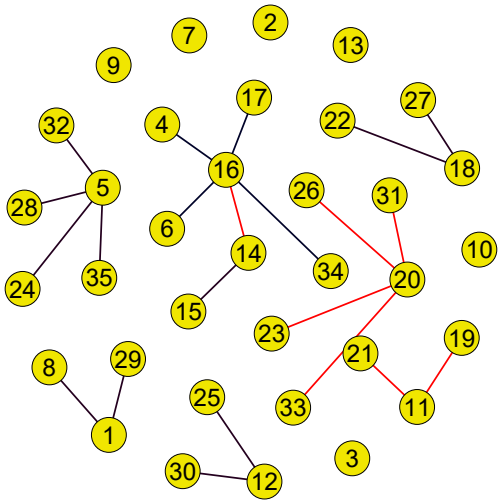
	usds	csúcsok
	151	4 6 17 34
▶	149	14 19 21 23 26 31 33
	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.11. ábra. *A Recursive eredménye az 5. lépés után; 2. elágazás a 18-ból a 2. szinten.*



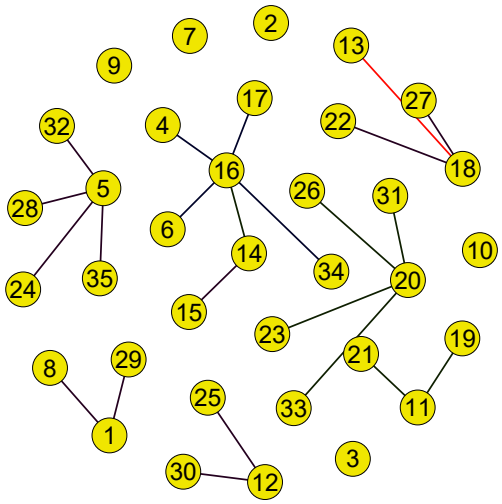
	usds	csúcsok
	151	4 6 17 34
	149	14 19 21 23 26 31 33
▶	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.12. ábra. *A Recursive eredménye az 6. lépés után. A 7. lépésben zsákutcához ér.*



	usds	csúcsok
	151	4 6 17 34
▶	149	14 19 21 23 26 31 33
	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

2.13. ábra. *A Recursive eredménye az 8. lépés után; 3. elágazás a 18-ból a 2. szinten.*

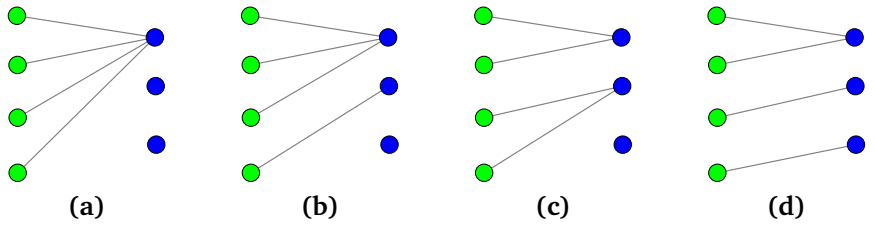


	usds	csúcsok
	151	4 6 17 34
	149	14 19 21 23 26 31 33
▶	141	18
	118	3 9 16 24 28 32 35
	116	7 10 11 20
	114	12
	112	13
	89	1
	86	2
	85	5

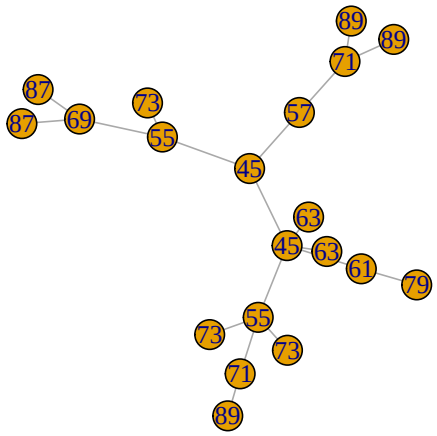
2.14. ábra. *A Recursive eredménye az 9. lépés után; zsákutcához vezet.*

```
1  Function GenerateDispositions(nodes, η, ds_neig)
2      dispSet := List()                          // empty list for subsets
3      for i in Unique(ds_neig) do
4          nodes_i := nodes[Where(ds_neig == i)]
5          η_i := η[Where(ds_neig == i)]
6          ns := Length(nodes_i)
7          ms := Length(η_i)
8          σ := IntegerSplitUps(ns, ms)
9          σ := Prune(G, σ, η_i[1] [, nodes_i[1]])
10         dispSet[i] := ∅
11         for s in σ do
12             dispSet[i] := dispSet[i] ∪ Edge(nodes_i, η_i, s)
13     return Combination(dispSet)
```

5. Algoritmus: Az élek lehetséges elhelyezésének gyártása



2.15. ábra. *Szeretnénk az összes lehetséges megoldást, ahol a bal oldalon szereplő (zöld) csúcsok mindegyikét a jobb oldalon lévő valamennyi (kék) csúcshoz rendelünk. Az összes elhelyezést láthatjuk, mikor $ns = 4$ és $ms = 3$. A részhalmazok az $fnode$ szomszédjainak száma szerint, amik a Vágó (Pruning) algoritmusokban jelennek meg, az {(a)}, {(b)}, és {(c), (d)}.*



2.16. ábra. *A csúcsok címkéi a ds értékek. A Recursive Solver futásának kezdetén néhány levél csúcsot a szomszédjához kötünk, 87 és 79 értékű csúcsok. Az $usds$ első három eleme a 89, 73 és 71. Mikor a $step = 3$ (az aktuálisan vizsgált csúcsok ds értéke 71) két csúcs van ugyanazzal a ds értékkel, de az összefüggő gráfjaik mérete különböző (3 és 2). Az algoritmus két különböző részhalmazba rakja ezeket a csúcsokat. Legyen $nodes_1$ a 3 méretű csúcsot tartalmazó részhalmaz, a $nodes_2$ pedig a 2 méretű csúcsot tartalmazó. Az η_1 egyelemű lesz (az 57 értékű csúcs), és η_2-nek két eleme lesz (az 55 értékű csúcsok).*

musokban ezt a csúcsot $fnode$-nak nevezzük. A σ-beli sorozatok nemnegatív egész értékeket csökkenő sorrendben tartalmaznak. A számok jelzik, hogy az η_i csúcsainak hány szomszédja van a $nodes_i$-ből. A sorozat első eleme a legnagyobb. Ezt a számot az $\eta_{max}[i][j]$-vel jelöljük. Ez a legnagyobb érték az η_i j-edik él elhelyezésében.

A vágó algoritmusok első lépésben a sorozatokat (vagy elhelyezéseket) csoportosítják a legnagyobb értékek szerint (η_{max} elemek). Egy η_i halmazban az első csúcsnak ($fnode$ a 6. és a 7. Algoritmusokban) ugyanazon szomszédjai vannak minden elhelyezésben, ami az η_i-hez tartozik. Ezeknek a szomszédoknak a száma $\eta_{max}[i][j]$. Az 5. Algoritmusban σ lesz a részhalmazok halmaza. Az algoritmusok (2.3)-t használva

```
1 Function Prune(G, σ, fnode)
2     σ_sets := DivideByFirstValue(σ)
3     for s in σ_sets do
4         asnodes := NodesFromAssignment(σ, s, fnode)
5         κ := EstimateNeighbourDS(G + Edge(asnodes, fnode), fnode)
6         if κ ∉ δ then
7             σ := σ \ Assignments(σ, s)
8     return σ
```

6. Algoritmus: Rough Pruning

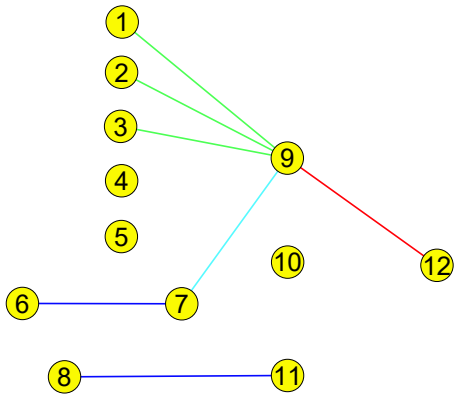
```
1 Function Prune(G, σ, fnode, nodes_i[1])
2     σ_sets := DivideByFirstValue(σ)
3     index := Where(ds[nodes_i[1]] < usds & usds < ds[fnode])
4     ξ := Where(δ == usds[index])    // possible lower value neighbours
5     for s in σ_sets do
6         vs := NodesFromAssignment(σ, s, fnode)
7         for j in P(ξ) do                          // power set of ξ
8             κ := κ ∪ EstimateNeighbourDS(G + Edge(vs ∪ j, fnode), fnode)
9         if ∀k ∉ δ : k ∈ κ then
10            σ := σ \ Assignments(σ, s)
11    return σ
```

7. Algoritmus: Soft Pruning

tesztelik, hogy η_i-ből az első csúcsnak ($fnode$) $nodes_i$-ből $\eta_{max}[i][j]$ szomszéddal van vagy nincs lehetséges kisebb ds értékű szomszédja.

– Rough Pruning: az első vágó metódus csak a $\eta_{max}[i][j]$ érték alapján dönt a vágásról. A sorozatokból részhalmazokat készítünk (σ_{sets}) és összegyűjtjük a csúcsokat ($asnodes$), amelyek az $fnode$-hoz tartoznak (6. Algoritmus). Új becslést csinálunk minden $\eta_{max}[i][j]$-hez ($fnode\ \eta_{max}[i][j]$ darab egyforma szomszéddal). Becsüljük az $fnode$ lehetséges szomszédjának ds értékét az összegyűjtött élek alapján. Jelöljük ezt a becsült értéket κ-val (6. Algoritmus 5. sora). Ha κ nincs a δ input vektorban, akkor vágjuk le ezt az elágazást. A 2.17. ábrán a metódus a megoldást is levágta, mert a 7-es csúcs a legnagyobb értékű csúcs (9-es csúcs) szomszédja, de a ds értéke kisebb, mint az aktuálisan vizsgált érték (1-5-ös csúcsok), és nem kötöttük őket össze a jelenlegi gráfban. Így a ds értéket (12-es csúcs értékét) rosszul becsültük.

– Soft Pruning: ebben a vágó metódusban nem dobjuk el azokat az eseteket, amiket a 2.17. ábrán láthatunk. Több számítást igényel és kevesebb elágazást vág



2.17. ábra. *Példa az elhelyezésekre (Recursive Solver-ben). A csúcsok vízszintes helyzete mutatja a ds értékeiket, balról jobbra csökken. Az 1-5-ös csúcsok tartoznak a jelenleg vizsgált ds értékhez. A 9-11-es csúcsoknak megegyezik az értéke a becslés eredményével. A 6-os és 8-as csúcsoknak nagyobb az értéke mint a vizsgált, a 7-esnek kisebb, de a becsültnél nagyobb. A 12-es csúcs egy lehetséges szomszédja a 9-esnek. Az aktuális él elhelyezésben az 1-3-as csúcsok a 9-es szomszédai, a 4-es és 5-ös a 10-es és 11-es szomszédai valamilyen módon. Az 1-5-ös csúcsok a $nodes_i$, 9-11-es csúcsok az η_i a 5. algoritmusban. A 9-es csúcs az $fnode$ a vágó algoritmusokban. A 7-es csúcs lehet szomszédja a 9-esnek, de ezt csak a vágások számításánál használjuk, nem része az él elhelyezésnek. A 7-es csúcs eleme a ξ-nek a 7. algoritmusban.*

le. A kerete hasonló az elsőéhez, de számításba veszi a csúcsok (mint a 7-es) becslésénél, hogy az $fnode$ (9-es csúcs) szomszédai lehetnek azok a csúcsok, amelyek ds értéke az aktuálisan vizsgált érték (1-es csúcs értéke) és az első becslés (a vágáson kívüli, 4. Algoritmus 10. sora) értéke (9-es csúcsé) között van. Összegyűjtjük ezeket a lehetséges szomszédokat a ξ változóba (7. Algoritmus 4. sora). A ξ-beli csúcsok a második becslésnél (a vágáson belül a 8. sor) fel lesznek használva, de ebben az iterációban nem kötjük őket csúcshoz a gráfon belül. Ezen csúcsok összes lehetséges kombinációja le lesz tesztelve (7. sor). Ha a második becslés értéke (12-es csúcsé) létezik a δ input vektorban (9. sor), akkor nem vágjuk le azokat az elhelyezéseket, amik az $\eta_{max}[i][j]$-hoz tartoznak.

2.5. Numerikus kísérletek

2.5.1. Számítógépes környezet

Az összes javasolt algoritmus R programozási nyelven lett implementálva az `igraph` [13] és a `BBmisc` könyvtárak használatával. A használt számítógép Intel Xeon E5-2660 (2.00 GHz) CPU-val és 64 GB memóriával rendelkezett, Ubuntu Linux 18.04.5

operációs rendszerrel. Habár az algoritmusaink több része párhuzamosítható, csak a következő részeket valósítottuk meg párhuzamos végrehajtással a tesztek elvégzéséig: az új lehetséges ds értékek számítása a heurisztikák döntéshozatalánál; az elhelyezések csoportosítása és a change-making-szerű probléma részek (külső for ciklus) a 5. algoritmusban; és a nem megfelelő elhelyezések kiválasztása a részhalmazokban rész (`for` ciklus) a vágó algoritmusokban. Egy másik párhuzamosítható rész az elágazás. A Recursive algoritmus jelenlegi állapotában az elágazások pontjait mélységi kereséssel járja be.

2.5.2. Adathalmaz

A javasolt algoritmusok teszteléséért véletlen gráfokat gyártottunk a Barabási-Albert modellel [2]. Ez a módszer az úgynevezett prefential attachment növekedési mechanizmust használja, ahol minél több kapcsolata van egy csúcsnak, annál nagyobb valószínűséggel kap új élet. Ezeknek a gráfoknak hatvány (*power law*) fokszám eloszlása van. Formálisan $p_k \sim k^{-\alpha}$, ahol p_k a k fokú csúcsok (várható) száma és $\alpha > 0$ egy paraméter.

Az α mellett az m egy másik paraméter, amely szabályozza a növekedést: az újonnan csatlakozó csúcsok m darab már meglévő csúcshoz kapcsolódnak. Fa gráfok készítéséhez az m paramétert 1-nek választottuk.

Két adathalmazt készítettünk a következő jellemzőkkel:

– Az első halmazt csak a Maximum Presolver és Corrector (2.4.2. alfejezetből) algoritmusokkal használtuk. A gráfok méretei a halmazban $20, 35, 50$ és 75, míg az α az $1, 1{,}8$ és $2{,}25$ értékekre lett választva. Az α értékeket követve 50, 30 és 30 darab készült minden mérethez. Minden mérethez készítettünk három kiegyensúlyozott gráfot (az ábrákon és tablázatokban Balanced-nak jelöltük), ahol a csúcsoknak $k = 2, 3$ vagy 4 gyereke van, kivéve a leveleknek és a levelek szüleinek. megjegyzendő, hogy egy csúcsnak lehet kevesebb gyereke, mint k, de a csúcs nem levél.

– A második halmazban a gráfok méretei megegyeznek, $N = 1000$, mindegyik a Barabási-Albert modellel készült. Az α értékek $1, 1.5$ és 2. Minden kategóriához 50 teszt gráfot alkottunk. Ez a halmaz a Maximum Presolver és a Recursive solver tesztelésénél volt használva.

2.5.3. Eredmények

2.5.3.1. Első adathalmaz

Futási idők. A 2.18. és 2.19. ábrán láthatjuk az eredeti (rekonstruálandó) fa gráf és az eredmény normalizált CC értékei közti átlagos eltérést minden kategóriában. Az

eltérés a két vektor különbségének Eklidészi-normájaként lett számolva. Szeretnénk megjegyezni, hogy az alapmérték meghatározásáért (baseline) összehasonlítottuk az eredeti fa gráfot a megfelelő csillag (Star) és út (Path) gráfokkal (a 2.18. és 2.19. ábrákon az első két oszlopcsoport). Ezekben az összehasonlításokban a vektorokat rendeztük, hogy a minimális eltérést kapjuk meg, az legközelebbi izomorf megoldástól való eltérést. A csillag és út gráfok az adott méretű gráfok csoportjának két szélsőséges esete, ezért választottuk ezeket az összehasonlítási alaphoz. Mivel a nagyobb α paraméter csillagszerűbb input gráfokat eredményez, nem meglepő, hogy a csillag gráfhoz egyre közelebb kerül a kívánt struktúra, míg az út gráfnál egyre nagyobb hibát látunk. Ezekben az alapmérték kísérletekben a Balanced gráfokra általában ugyanolyan gyenge eredményeket kaptunk, mint az $\alpha = 1$ esetekre.

A négy Corrector eredményének összehasonlításakor is a normalizált CC vektorokat használtuk, de más módosítást nem csináltunk (például rendezést).

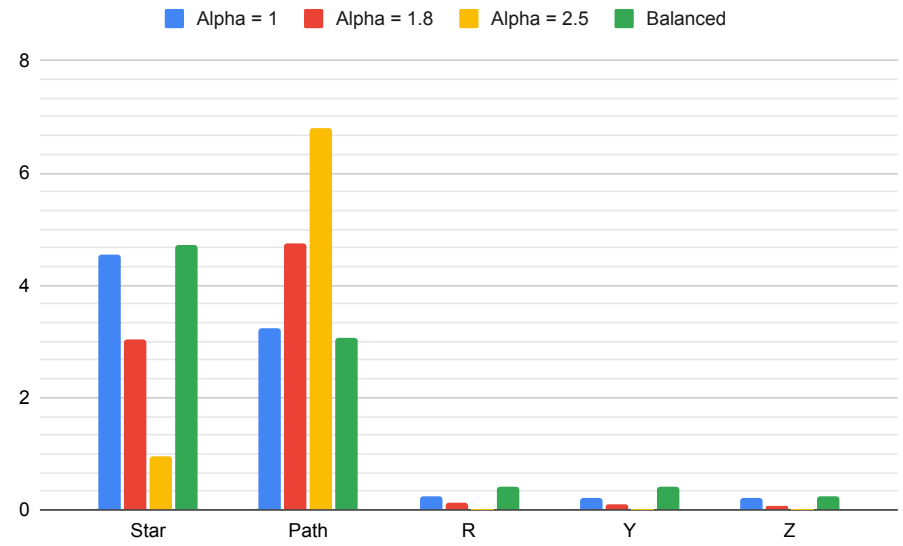
Az F-Corrector, amelyet a 2.4.2. alfejezetben mutattunk be, olyan gyenge eredményeket produkált az összes előteszten, hogy úgy döntöttünk a Corrector ezen verziójának vizsgálatát nem folytatjuk. Ezért az F-Corrector eredményei nem szerepelnek az ábrákon. A gyenge eredmények azzal a ténnyel magyarázhatóak, hogy ez a heurisztika nem tud kilépni a lokális optimumokból (egy lépésben próbálja optimalizálni a teljes különbség vektort), míg a többi javító algoritmus (Z, Y, R verziók) csak egy értékre összpontosít és látszólag rossz lépéseket csinál, amelyek képesek kilépni egy lokális megoldás vonzáskörzetéből. Hasonlóan az ismert Szimulált Hűtés (Simulated Annealing) [39] koncepciójához, a globális optimum eléréséért előnyös néha a legrosszabb megoldások felé haladni. Így a javító algoritmusnak érdemes kevésbé mohó típusúnak lenni.

Az átlagos eltérés az $\alpha = 2.5$ kategóriákban bármely Corrector algoritmussal majdnem mindig 0.

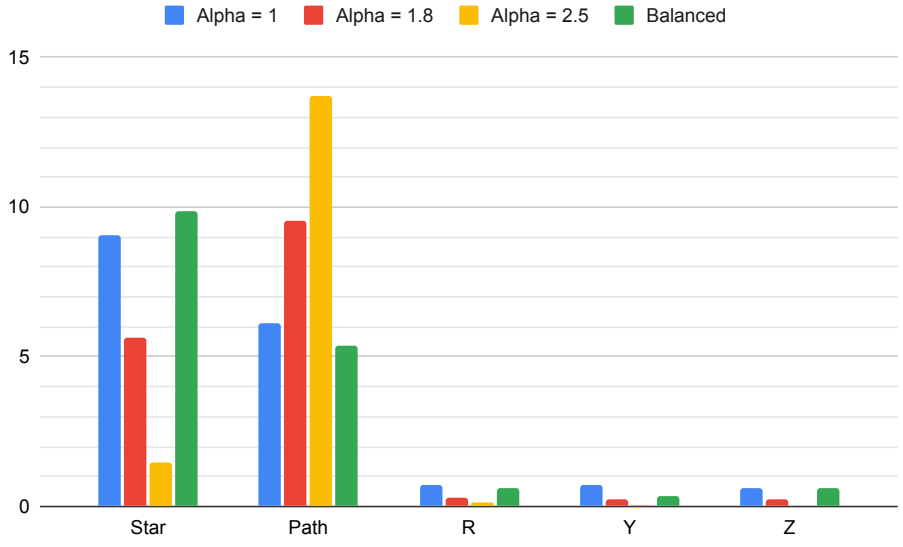
Sikerarányok. A javasolt algoritmusok sikerességére vonatkozó részletesebb eredményeket a 2.3. táblázat tartalmazza. Általánosságban az $\alpha = 1$ esetek voltak a legnehezebbek. Nagyobb problémáknál a bukás aránya (amikor egyik algoritmus sem tudta megtalálni a pontos megoldást) 80% körül volt. Magasabb α paramétereknél, akárcsak a kiegyensúlyozott gráfoknál, a sikerek aránya szinte tökéletes volt. Kiemelnénk, hogy a legtöbb eseten, az $\alpha = 2.5$-nél kifejezetten, a Maximum Presolver elég hatékony volt. Az R-Corrector adta a legrosszabb eredményeket minden kategóriában.

2.5.3.2. Második adathalmaz

A második adathalmazon végzett teszteknél egy futás 'bukottnak' lett jelölve, ha a Maximum Presolver és egy Correcot nem tudta elérni a globális optimumot $(N - -1)/2$ iterációs lépésig, vagy a Recursive solver '`Not Solved`' értéket adott (azaz az



(a)

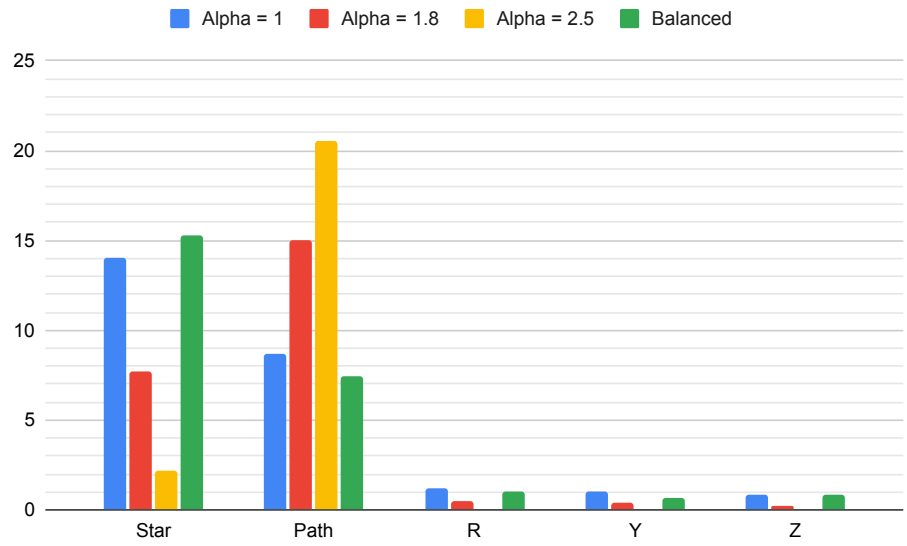


(b)

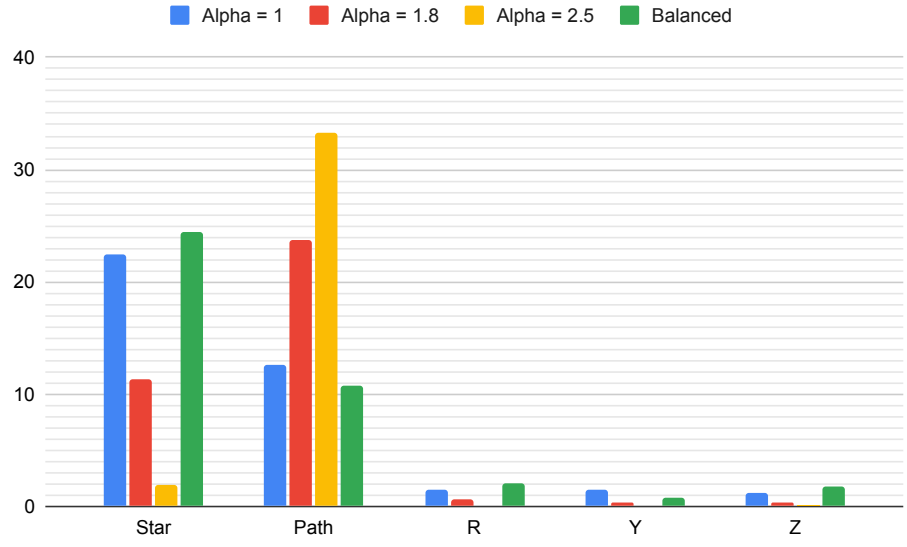
2.18. ábra. *Átlagos eltérés a kategóriákban: (a)* $N = 20$*, (b)* $N = 35$

összes elágazást megvizsgálta és nem talált megfelelő megoldást - a Rough Pruning kivágta a megoldást) vagy nem fejeződött be 10 percen belül.

Sikerarányok. A javasolt algoritmusok sikereinek eredménye a második adathalmazon való végrehajtásból a 2.4. táblázaton látható. Egyértelműen egyik algoritmus sem tudta megoldani az $\alpha = 1$ eseteket. A másik oldalon a Maximum Presolver megoldotta az összes $\alpha = 2$ paraméterű feladatot, a Corrector algoritmusok segítsége



(a)



(b)

2.19. ábra. *Átlagos eltérés a kategóriákban: (a)* $N = 50$*, (b)* $N = 75$

nélkül. Az $\alpha = 1.5$-ös feladatok körülbelül fele bizonyult nehéznek, a Maximum Presolver 24%-át oldotta meg ezeknek. A Maximum Presolver által elbukott eseteken a Z-Corrector volt a leghatékonyabb, a teszt problémák további 24%-ának megoldásával.

A Recursive solver eredményei a 2.5. táblázatban szerepelnek. Az $\alpha = 1$ esetek nehezek voltak ennek az algoritmusnak is, egyet sem tudott megoldani közülük. A

Rough vágó metódussal az összes $\alpha = 2$ problémánál sikeres volt. Az $\alpha = 1.5$ eseteknél a Rough vágással a problémák 58%-a lett megoldva, és volt egy, amelyet a Soft vágással oldottuk meg, de a Rough nem tudott.

A közölt eredmények alapján azt mondhatjuk, hogy a legváltozatosabb eredményeket az $\alpha = 1.5$ esetekre kaptuk. Ennélfogva, a 2.6. táblázatban összehasonlítottuk a teszt eseteket, hogy (i) hányszor volt képes a Maximum Presolver a Z-Corrector-ral és a Recursive a Rough vágó metódussal megoldani a problémákat (az esetek 40%-a), illetve (ii) hányszor sikerült legalább egyiküknek megtalálni a globálisan optimális megoldást (72%).

Futási idők. A Maximum Presolver-nek Z-Corrector-ral és Recursive algoritmusnak a Rough vágóval való sikeres futási idejei a 2.7. táblázatban vannak összehasonlítva.

Gráfok tulajdonságai. A második adathalmazban lévő és az algoritmusaink által rekonstruált gráfok átmérőjét és a bennük szereplő távolságok átlagát megmértük. Az

2.3. táblázat. *Az első adathalmaz eredményei. Az oszlopok jelölései a következőket jelentik. Nem: azon esetek száma, mikor egyik megoldó sem érte el a megoldást; Presolve: a Maximum Presolver által megoldott esetek száma, a javító fázisra nem volt szükség; R: a Presolver nem oldotta meg, csak az R-Corrector; Y: a Presolver nem oldotta meg, csak az Y-Corrector; Z: a Presolver nem oldotta meg, csak az Z-Corrector; Z+: a Presolver nem oldotta meg, a Z-Corrector és a többi heurisztika közül egy igen; All: Presolver nem oldotta meg, de az összes javító heurisztika igen.*

méret	kitevő	Nem	Presolve	R	Y	Z	Z+	Mind	#gráfok
$N = 20$	$\alpha = 1$	13	17	0	2	2	7	8	50
	$\alpha = 1.8$	2	22	0	1	0	2	3	30
	$\alpha = 2.5$	0	30	0	0	0	0	0	30
	Balanced	0	1	0	0	0	2	0	3
$N = 35$	$\alpha = 1$	40	4	0	0	0	5	1	50
	$\alpha = 1.8$	3	16	0	1	1	0	9	30
	$\alpha = 2.5$	0	28	0	0	0	1	1	30
	Balanced	1	1	0	0	0	1	0	3
$N = 50$	$\alpha = 1$	41	1	0	0	4	2	2	50
	$\alpha = 1.8$	6	16	0	0	0	4	4	30
	$\alpha = 2.5$	0	29	0	0	0	0	1	30
	Balanced	2	1	0	0	0	0	0	3
$N = 75$	$\alpha = 1$	44	1	0	1	2	1	1	50
	$\alpha = 1.8$	3	16	0	2	1	4	4	30
	$\alpha = 2.5$	0	29	0	0	0	0	1	30
	Balanced	2	1	0	0	0	0	0	3

2.4. táblázat. *A Maximum Presolver és a Corrector algoritmusok eredményei a második adathalmazon. Az oszlopok megegyeznek a 2.3. táblázatéival.*

méret	kitevő	Nem	Presolve	R	Y	Z	Z+	Mind	#gráfok
$N = 1000$	$\alpha = 1$	50	0	0	0	0	0	0	50
	$\alpha = 1.5$	23	12	0	0	12	2	1	50
	$\alpha = 2$	0	50	0	0	0	0	0	50

2.5. táblázat. *A második adathalmazon végrehajtott Recursive solver eredményei. Az oszlopok jelentései: Nem = az algoritmus* `'Not Solved'` *üzenettel állt meg; Rough = a Rough vágás metódussal megoldottak; Soft = a Rough nem oldotta meg, de a Soft igen.*

$N = 1000$	Nem	Rough	Soft	#gráfok
$\alpha = 1$	50	0	0	50
$\alpha = 1.5$	20	29	1	50
$\alpha = 2$	0	50	0	50

eredmények a 2.8. táblázatban mutatjuk meg. A Recursive algoritmus csak akkor tér vissza gráffal, ha pontos eredményt talált. Az $\alpha = 1$-es és $\alpha = 1.5$-ös gráfok közül volt, amit a Recursive nem oldott meg. A Recursive által gyártott gráfok mindegyikének értéke megegyezik az eredeti párjával. Az eltérést a hiányzó eredmények okozzák.

A levelek korlátolt ismerete. Az átlagos levél mennyisége a teszt fa gráfoknak $\alpha = 2$-nél 991.76 míg $\alpha = 1.5$-nél 930.9 volt. Hogy lássuk mennyire fontos, hogy ismerjük mely csúcsok a fa levelei, teszteket végeztünk a Recursive algoritmussal Rough vágással az $\alpha = 2$ gráfokon, ahol az ismert levelek számát 900-ra korlátoztuk. Az átlagos futási idő 0.3546 másodpercre növekedett. Szintén teszteltük ezeket a gráfokat a Recursive algoritmussal, mikor csak a legnagyobb ds értékű levelek közül egyet ismertünk. Az átlagos futási idő tovább nőtt, 0.4402 másodperc. Ezekben a tesztekben a megoldhatóság megegyezett az előzőekkel, minden futás sikeres volt. Ez azt jelenti,

2.6. táblázat. *Összesített eredményei a Maximum Presolver-nek a Z-Corrector-ral (Pre+Z) és a Recursive-nak a Rough vágóval (RecRough) a második adathalmaz $\alpha = 1.5$ esetein. Oszlopok jelentései: Nem = egyik algoritmus sem oldotta meg pontosan; Pre+Z ∩ RecRough = mindkettő megoldotta; Pre+Z ∪ RecRough = legalább egy adott gráfot a kívánt ds értékekkel.*

$N = 1000$	Nem	Pre+Z ∩ RecRough	Pre+Z ∪ RecRough	#gráfok
$\alpha = 1.5$	14	20	36	50

2.7. táblázat. *A második adathalmaz sikeres eseteinek átlagos futási ideje. Maximum Presolver Z-Corrector-ral (Pre+Z) és a Recursive a Rough vágóval (RecRough).*

$N = 1000$	Sikeres Pre+Z átlagos ideje	Sikeres RecRough átlagos ideje
$\alpha = 1.5$	0.669 sec	6.148 sec
$\alpha = 2$	0.098 sec	0.089 sec

2.8. táblázat. *A második adathalmaz gráfjainak és a maximum Presolver Z-Corrector-ral illtve Recursive Rough vágóval való eredmény gráfjainak átlagos mérései.*

$N = 1000$		Eredeti gráfok	Pre+Z eredményei	RecRough eredményei
$\alpha = 1$	Átlagos távolság	6.8466	6.8891	-
	Átmérő	17.52	18.5	-
$\alpha = 1.5$	Átlagos távolság	2.3658	2.4161	2.3915
	Átmérő	8.4897	9.0204	8.3448
$\alpha = 2$	Átlagos távolság	2.0370	2.0370	2.0370
	Átmérő	4.88	4.88	4.88

hogy ezeknek a gráfoknak a megoldhatósága nem csak azon a többlet információn múlik, hogy néhány csúcs éleinek számát ismerjük, hogy tudjuk mely csúcsok levelek. A ds értékek a problémákban kevesebb lehetséges szomszédot generálnak, kevesebb elágazást.

2.6. Konklúzió

Ebben a munkában a gráf rekonstrukció közelség központisági értékekből probléma egy specifikált verzióját vizsgáltuk. A specifikációk, hogy a rekonstruálandó gráf egy súlyozatlan fa, a közelség központiság reciprok értékeit (ds értékek) használtuk, ezek a ds értékek egészek, mert a gráf súlyozatlan és többlet információt használtunk, amely azt mondta meg, mely csúcsok levelek.

Három algoritmust javasoltunk a specifikus probléma megoldására. Az első algoritmus, Maximum Presolver, célja egy közelítő megoldás megtalálása gyorsan. Habár ennek az algoritmusnak az eredmény gráfja gyakran nem a pontos megoldás, az használható más algoritmusok jó kezdő pontjaként (*warm start*). Ezt használja a második algoritmusunk, a Corrector. Ez az algoritmus egy metaheurisztika és négy verzióját alkottuk meg. A harmadik algoritmus, Recursive solver, a pontos megoldás megtalálására készült. A három algoritmus közül ennek a legnagyobb az időkomplexitása. Azt a tételt használja, amely megmondja mik lehetnek két szomszédos csúcs ds ér-

tékei. Feltételének egyes részei (a szomszédos csúcsok egyikéhez a gráfban közelebb helyezkedő csúcsok száma) nem mindig ismertek. Ezen részek korlátain belüli összes értéket kipróbálja a tétel felhasználásához. Ezen okból az algoritmus elágazhat. Két vágó algoritmust készítettünk a Recursive solver felgyorsításáért.

A Barabási-Albert metódussal készített fa gráfokat teszteltünk. A magasabb α paraméterű problémák sikeresebbnek bizonyúltak minden algoritmusunk esetében. A 2.5. alfejezet végén megmutattuk, hogy a magasabb α-val rendelkező gráfok sikeressége nem csak a levelek létéről szóló többlet információnak köszönhető.

A ds vektorból (δ input vektor) való levelek detektálása és ezen információk átadása az algoritmusoknak vagy az algoritmusok fejlesztése a kevesebb extra adat használásáért jövőbeli munkánk. A bemutatott algoritmusok segíthetnek megoldani a Közelségi központiságból gráf rekonstruálhatóság problémáját (graph realization) és az általános egyszerű gráf rekonstrukció Közelségi központiságból problémát.

Néhány, a Bevezetésben említett kapcsolódó munka sztochasztikus vagy populáció alapú algoritmusokat használ más centralitási mértékekhez. Módosítani és tesztelni őket CC-re egy jövőbeni munkánk lehet.

3. fejezet

Gráf realizáció

Az előző fejezetben vizsgált rekonstrukció problémához kapcsolódó feladatot tárgyalunk. A realizáció során a gráfot nem kell meghatározni, csak eldönteni, hogy létezik-e. Ennek érdekében szükséges feltételeket határoztunk meg fa és nem-fa gráfok közelség centralitási értékeire (pontosabban azok reciprokaira).

3.1. Bevezető

A feladat egy N hosszú vektorról eldönteni, hogy egy N csúcsú összefüggő G gráf csúcsonkénti CC értékeit tárolhatja-e. Pontosabban az, hogy kiszűrjük azokat az N hosszú vektorokat, amikhez garantáltan nem lehet egy G gráfot rendelni, ahol a G i-edik csúcsának CC értéke a vektor i-edik eleme.

Az előző fejezethez hasonlóan a CC értékek helyett az egész értékű reciprokaikat használjuk, amelyeket ds értékeknek nevezünk. Így a kérdéses vektorban is a ds értékek szerepelnek.

Két csoportra sikerült szükséges feltételeket megfogalmazni: egyszerű, irányítatlan, súlyozatlan, összefüggő gráfokon belül fákra illetve nem-fákra.

Amennyiben ezen feltételek nem teljesülnek, akkor az előző fejezetben tárgyalt vagy hasonló költséges algoritmusok végrehajtását elhagyhatjuk. Sajnos olyan vektorok is teljesítik ezeket a feltételeket, amelyekhez nem tudunk gráfot rendelni az említett módon. Ezek a feltételek nem elégségesek.

A [20] cikkben a fokszámokra készítettek szükséges és elgendő feltételt.

3.2. Feltételek fákra

3.1. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, fa gráf esetén, ha $|V| = N$, akkor $2 \mid \sum_{v \in V} ds(v)$. Ha $2 \mid N$, akkor $\forall v : \ (2 \mid ds(v) \vee 2 \nmid ds(v))$.*

Bizonyítás. Minden út ugyanannyival növeli a két végcsúcs ds értékét. Az összeg osztható lesz 2-vel. K_2 gráfnál a $\sum_v ds(v) = 2$, $2 \mid 2$. Ha új csúcsot adunk a gráfhoz, akkor minden régi csúcs ds értéke összesen annyival növekszik, mint az új csúcs ds értéke. Így, ha az új csúcs v_n, akkor az új gráf csúcsainak ds összege egyenlő $\sum_{v \in V, v_n \notin V} ds(v) + + 2 \cdot ds(v_n)$. Mivel

$$2 \mid \sum_{v \in V, v_n \notin V} ds(v) \text{ és } 2 \mid 2 \cdot ds(v_n), \text{ így } 2 \mid \sum_{v \in V, v_n \notin V} ds(v) + 2 \cdot ds(v_n).$$

Triviális, hogy a K_2 gráfnál minden csúcs ds értéke 1 (3.1. ábra (a) részének bal oldala).

Ha a $G_1 = (V_1, E_1)$ gráfhoz, ahol $2 \mid N_1$ és minden csúcs ds értéke páratlan, hozzáadunk egy új v_y csúcsot, amit a már gráfban szereplő v_x csúcshoz kötünk, akkor az új $G_2 = (\{V_1, v_y\}, \{E_1, (v_x, v_y)\})$ gráfban $ds_{G_2}(v_x)$ páros lesz, mivel $ds_{G_2}(v_x) = = ds_{G_1}(v_x) + 1$. Az új csúcs 1 távolságra lesz az v_x-től. A 2.1. Tétel szerint $ds_{G_2}(v_y) = = ds_{G_2}(v_x) + (N_2 - 2)$. Mivel $2 \mid ds_{G_2}(v_x)$ és $2 \nmid (N_2 - 2)$, így $2 \nmid ds_{G_2}(v_y)$. A v_y-tól páros távolságra lévő csúcsok ds értéke páros számmal növelődik. Páratlanok voltak G_1-ben, így páratlanok is lesznek G_2-ben. A páratlan távolságra lévő csúcsok új ds értéke páros lesz.

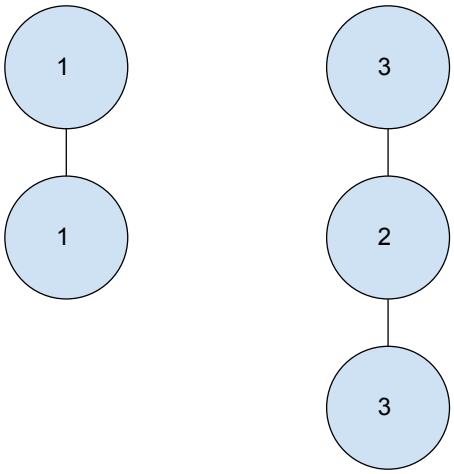
Minden páros ds-sel rendelkező csúcs szomszédjának ds értéke páratlan és minden páratlan ds értékű csúcs szomszédjának ds értéke páros a G_2-ben (lásd 3.1. ábra (a) részének jobb oldala).

Ha G_2 gráf csúcsainak száma páratlan ($2 \nmid N_2$) és az előbb említett tulajdonság teljesül (szomszédos csúcsok ds értékeinek paritása különböző), akkor egy új v_z csúcs hozzáadásával készítjük G_3-at és a következők lesznek az értékek:

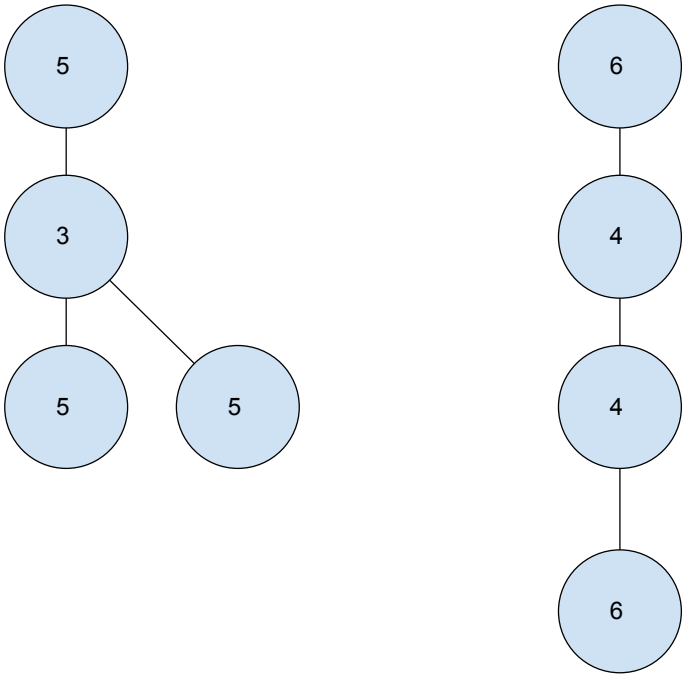
1. v_z csúcsot egy $v \in V(G_2)$ csúcshoz kötjük, ahol $2 \mid ds_{G_2}(v)$. Ekkor $ds_{G_3}(v) = = ds_{G_2}(v) + 1$ és $ds_{G_3}(v_z) = ds_{G_3}(v) + (N_3 - 2)$. Mivel $ds_{G_2}(v)$ páros, így $2 \nmid ds_{G_3}(v)$. Az N_3 páros, így $2 \nmid ds_{G_3}(v_z)$. Minden v_z-től páros távra lévő csúcs eredeti ds értéke páratlan volt, mivel v-é páros volt. A páratlan ds értékekhez páros számot adva páratlant kapunk. A páratlan távra lévők értékei párosak voltak, így azok szintén páratlanok lesznek. Minden csúcs ds értéke páratlan G_3-ban (lásd 3.1. ábra (b) részének bal oldala).

2. v_z csúcsot egy $v \in V(G_2)$ csúcshoz kötjük, ahol $2 \nmid ds_{G_2}(v)$. Ekkor $ds_{G_3}(v)$ páros lesz, és $ds_{G_3}(v_z)$ is, hiszen $2 \mid (N_3 - 2)$. A v_z-től páros távra lévő csúcsok ds értéke páros volt, páros is marad. A páratlan távra lévők ds értéke páratlannal nől, így ezek új értékei párosak lesznek. Minden csúcs páros ds értékkel rendelkezik G_3-ban (3.1. ábra (b) részének jobb oldala).

Ha az első esetbeli G_3-at bővítenénk, akkor G_1 esetét ismételjük és kapunk G_2-höz hasonló gráfot. Ha a második esetbeli G_3-hoz adunk új csúcsot, akkor az új csúcstól



(a)



(b)

3.1. ábra. *Csúcsok ds értékeinek változása a gráf méretének növelése során*

páratlan távra lévők ds értéke páratlan lesz. Minden páros távra lévő új értéke páros marad. Az új csúcs ds értéke páros lesz, mivel a szomszédjáé páratlan és $(N-2)$-vel nagyobb, ahol $2 \nmid N$. Így pedig a G_2 esethez tértünk vissza. □

3.2. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, fa gráf minden v csúcsára igaz, hogy $N - 1 \leq ds(v) \leq \frac{N(N+1)}{2}$, ahol $|V(G)| = N$.*

Bizonyítás. A $ds(v)$ értékére az alsó és felső korlátokat külön-külön bizonyítjuk.
Alsó korlát:

1. Legyen $G = S_{N-1}$, ekkor a gráf egy v_x csúcsához pontosan $N-1$ darab 1 hosszú út kapcsolódik, más nem. A többi csúcshoz 1 darab 1 hosszú és $N-2$ darab 2 hosszú út. A gráfban v_x csúcsnak lesz a legkisebb ds értéke, ami $N-1$.

2. Ha $G \neq S_{N-1}$, akkor bármely v_x csúcsba az $N-1$ darab csúcsból legfeljebb $N-2$ darab 1 hosszú út vezet és legalább 1 darab legalább 2 hosszú út. Minden út, ami nem 1 hosszú, az legalább 2. Így $\forall v_x : \ ds(v_x) \geq N - 2 + 2 = N$.

Felső korlát:

1. Legyen $G = P_N$ és legyen $v_x \in V(G)$, $\deg(v_x) = 1$. Ekkor $ds(v_x) = N(N-1)/2$, mivel az $N-1$ db egyéb csúcsból 1, 2, $\ldots N-1$ hosszú utak vezetnek v_x-be. Legyen $v_y \in V(G), \deg(v_y) = 2$, ekkor v_y-ba vezető utak 1, 2, $\ldots k$, $\left(\frac{N-1}{2} \leq k \leq N-2\right)$ és 1, 2, $\ldots N-1-k$ hosszúak. Így

$$\begin{aligned} ds(v_y) &= \frac{k(k+1)}{2} + \frac{(N-1-k)(N-k)}{2} \\ &< \frac{k^2+k+(N-k)^2}{2} = \frac{N^2+2k^2+k-2Nk}{2}. \end{aligned}$$

   Mivel $2k^2 - 2Nk < 0$ és $N > k$, ezért

$$\frac{N^2+2k^2+k-2Nk}{2} < \frac{N^2+k}{2} < \frac{N^2+N}{2}.$$

   Így $ds(v_y) < \frac{N(N+1)}{2}$.

2. Ha $G \neq P_N$, akkor létezik P_i részgráfja, ahol $i \leq N-1$ és i a lehető legnagyobb. Ekkor ha $v_x \in V(G)$ és $ds_G(v_x) = \max(\delta_G)$, akkor

$$ds_G(v_x) = \frac{i(i-1)}{2} + \sum_{v_y \in V(G) \setminus V(P_i)} d(v_x, v_y) < \frac{i(i-1)}{2} + \sum_{j=i+1}^{N} j.$$

A P_N-ben nem megfeleltethető csúcsok távolabb helyezkednek el, mint a G-nek P_i-ben nem szereplő csúcsai v_x-től, hiszen P_i a lehető legnagyobb lineáris részgráfja G-nek. Mivel P_N-ben $\max(\delta_{P_N}) = \frac{N(N+1)}{2} = \frac{i(i+1)}{2} + \sum_{j=i+1}^{N} j$ tetszőleges i-re ($1 \leq i < N$), így $\max(\delta_{P_N}) > \max(\delta_G)$.

Ezek alapján $\forall v: \ ds(v) \leq \frac{N(N+1)}{2}$. □

3.3. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, fa gráfra igaz, ha δ a G gráf ds értékeinek vektora és $|V(G)| = N$, akkor* $N - 2 \leq \max(\delta) - \min(\delta)$.
$2 \mid N$ *esetén* $\max(\delta) - \min(\delta) \leq \frac{N^2}{4} - \frac{N}{2}$, *különben* $\max(\delta) - \min(\delta) \leq \frac{N^2}{4} - \frac{N}{2} + \frac{1}{4}$.

Bizonyítás. A $\max(\delta) - \min(\delta)$ különbség alsó és felső korlátját külön-külön bizonyítjuk.
Alsó korlát:

1. S_{N-1}-ben a különbség $N - 2$, mert a levelek ds értékei $2 \cdot (N - 2) + 1$ (2 hosszú utak és a középponttól való táv), a középső csúcs értéke $N - 1$ (1 távra van tőle minden rajta kívüli csúcs).

2. Ha $G \neq S_{N-1}$ és $v_m \in V(G), \ ds_G(v_m) = \min(\delta_G)$, akkor létezik $v_x \in V(G)$, hogy $d_G(v_x, v_m) = 2$.
   Legyen $v_y \in V(G)$, hogy $d_G(v_y, v_x) = d_G(v_y, v_m) = 1$. Legyen rendre j és k az elemszáma a $J = \{v \in V(G) : d_G(v, v_m) < d_G(v, v_x)\}$ valamint a $K = \{v \in \in V(G) : d_G(v, v_m) > d_G(v, v_x)\}$ halmazoknak.
   A v_m és v_x csúcs ds értékeihez ugyanannyit adnak azok a csúcsok, akik egyenlő távra vannak tőlük. Legalább egy ilyen csúcs van, v_y. Ezen csúcsok száma $N - - j - k$. A J halmazba tartozó csúcsok (j db) mindegyikéből 2-vel hosszabb út vezet v_x-be, mint v_m-be. Hasonlóan, a K halmazban szereplőktől 2-vel hosszabb utak (k db) vezetnek v_m-be, mint v_x-be. Így $ds_G(v_x) - ds_G(v_m) = 2j - 2k$.
   Mivel $j > N - j$ (mert $ds_G(v_m) < ds_G(v_y)$), így $j > \frac{N}{2}$, ezért $ds_G(v_x) - ds_G(v_m) > > N - 2k$, $k \geq 2$. Tehát $ds_G(v_x) - ds_G(v_m) > N - 2$, így

   $$\max(\delta_G) - ds_G(v_m) = \max(\delta_G) - \min(\delta_G) > N - 2,$$

   mert $\max(\delta_G) \geq ds_G(v_x)$.

Felső korlát:
P_N gráfban a maximális ds érték az $N(N-1)/2$. 1-től $N-1$-ig tartó sorozattal egyezik meg a levélbe vezető utak hosszai, többi csúcsnak a ds értéke ennél kisebb, hiszen 1-től i-ig tartó sorozat $\left(\frac{N}{2} \leq i < N - 1\right)$ és $N - i$ darab i vagy kevesebb értékű számok összegével egyeznek meg. A minimális érték két sorozat összegével egyezik meg. A

két sorozat összegei a $\frac{\left(\lfloor\frac{N-1}{2}\rfloor+1\right)\lfloor\frac{N-1}{2}\rfloor}{2}$ és $\frac{\left(\lceil\frac{N-1}{2}\rceil+1\right)\lceil\frac{N-1}{2}\rceil}{2}$. Így a minimális ds érték

$$\frac{\lfloor\frac{N-1}{2}\rfloor^2+\lfloor\frac{N-1}{2}\rfloor+\lceil\frac{N-1}{2}\rceil^2+\lceil\frac{N-1}{2}\rceil}{2}=\frac{\lfloor\frac{N-1}{2}\rfloor^2+\lceil\frac{N-1}{2}\rceil^2+N-1}{2}.$$

Ha $2 \mid N$, akkor $\lfloor\frac{N-1}{2}\rfloor+1=\lceil\frac{N-1}{2}\rceil=\frac{N}{2}$, különben $\lfloor\frac{N-1}{2}\rfloor=\lceil\frac{N-1}{2}\rceil=\frac{N-1}{2}$. Ezért

- $2 \mid N$ estén a minimális ds érték $\frac{\left(\frac{N}{2}-1\right)^2+\frac{N}{2}^2+N-1}{2}=\frac{N^2}{4}$.
- A $2 \nmid N$ estén pedig $\left(\frac{N-1}{2}+1\right)\frac{N-1}{2}=\frac{N^2-1}{4}$.

Adódik, hogy P_N-ben $\max(\delta)-\min(\delta)$ értéke

- $2 \mid N$ esetén $\frac{N(N-1)}{2}-\frac{N^2}{4}=\frac{N^2}{4}-\frac{N}{2}$.
- $2 \nmid N$ esetén $\frac{N(N-1)}{2}-\frac{N^2-1}{4}=\frac{N^2}{4}-\frac{N}{2}+\frac{1}{4}$.

Legyen most $G \neq P_N$, $v_m, v_n, v_x, v_{y_i} \in V(G)$, $(v_m,v_x),(v_m,v_{y_i}),(v_x,v_{y_i}) \notin E(G)$, $v_n \in \text{path}(v_m,v_x)$, $(v_m,v_n) \in E(G)$. Legyen $ds(v_m)=\min(\delta_G)$, $ds(v_x)>ds(v_m)$, $V_m=\{v : d(v,v_m)<d(v,v_n)\}$, $V_n=\{v : d(v,v_m)>d(v,v_n)\}$. Ekkor $|V_m|+|V_n|=N$, $|V_m| \leq |V_n|$.
Legyen $\forall v_{x'} : d(v_m,v_x) \geq d(v_m,v_{x'}), v_{x'} \in V_n$, $v_{z_i} \in \text{path}(v_m,v_x)$, $v_{y_i} \notin \text{path}(v_m,v_x)$ $(i \in \mathbb{N}^0)$, $(v_{z_i},v_{y_i}) \in E(G)$ és $V_{y_i}=\{v : d(v,v_{y_i})<d(v,v_{z_i})\}$, így $V_{y_i} \subset V_n$. Ekkor

$$ds(v_m)-ds(v_x)=d(v_m,v_x)\cdot|V_m|+\sum_i|V_{y_i}|\cdot(d(v_{y_i},v_x)-d(v_{y_i},v_m)),$$

mivel v_x-hez minden V_m-beli csúcsból $d(v_m,v_x)$-szel hosszabb út vezet, mint v_m-be, a $\text{path}(v_m,v_x)$-beli csúcsok v_m és v_x ds értékeihez végösszegben ugyanannyival járulnak, a V_{y_i} halmazokban lévő csúcsok elemenként $d(v_{y_i},v_x)-d(v_{y_i},v_m)$ járulnak többel ($d(v_{y_i},v_x)-d(v_{y_i},v_m)$ lehet negatív) a $ds(v_x)$-hez, mint a $ds(v_m)$-hez.

Mivel $|V_m|=N-|V_n|$, $|V_n|=d(v_m,v_x)-1+\sum_i|V_{y_i}|$ és teljesül, hogy $\forall i : d(v_{y_i},v_x)+$ $+d(v_{y_i},v_m)-2=d(v_m,v_x)$ (v_{y_i} szomszédja egy $\text{path}(v_m,v_x)$-beli csúcsnak), így

$$d(v_m,v_x)\cdot|V_m|+\sum_i|V_{y_i}|\cdot(d(v_{y_i},v_x)-d(v_{y_i},v_m))=$$

$$d(v_m,v_x)\cdot(N-d(v_m,v_x)+1-\sum_i|V_{y_i}|)+\sum_i|V_{y_i}|\cdot(d(v_m,v_x)-2\cdot d(v_{y_i},v_m)+2)=$$

$$d(v_m,v_x)\cdot(N+1-d(v_m,v_x))+\sum_i|V_{y_i}|\cdot(2-2\cdot d(v_{y_i},v_m)).$$

$N+1-d(v_m,v_x)>0$, így $d(v_m,v_x)\cdot(N+1-d(v_m,v_x))$ akkor és csak akkor maximális, ha $d(v_m,v_x)$ maximális.

Mivel $\forall i: \ d(v_{y_i}, v_m) = 0 \iff |V_{y_i}| = 0$ és $\forall i: \ d(v_{y_i}, v_m) > 2 \vee d(v_{y_i}, v_m) = 0$, így $\sum_i |V_{y_i}| * (2 - 2 \cdot d(v_{y_i}, v_m)) \leq 0$.
Ha $\sum_i |V_{y_i}| = 0$, akkor $d(v_m, v_x)$ maximális. Ezáltal adott $N, |V_n|$ esetén $ds(v_x) - - ds(v_m)$, akkor maximális, ha $d(v_m, v_x)$ maximális.

Előzőek szerint $d(v_m, v_x)$ akkor lesz maximális, ha $d(v_m, v_x) = |V_n|$. Ekkor $ds(v_x) - - ds(v_m) = |V_n| \cdot |V_m|$. Mivel $|V_m| + |V_n| = N$, így $|V_n| \cdot |V_m|$, akkor maximális, ha $|V_n| = |V_m|$. Ha $2 \nmid N$, akkor $|V_n| = |V_m| - 1$, mivel $|V_n| \leq |V_m|$-nek teljesülnie kell.
$|V_n| = |V_m|$ vagy $|V_n| = |V_m| - 1$ esetén $ds(v_x) = \max(\delta)$. Így $\max(\delta) - \min(\delta)$ akkor maximális, ha $|V_n| = |V_m|$ vagy $|V_n| = |V_m| - 1$.

A V_m csúcsokból álló részgráf szerkezete nem befolyásolja a különbséget. A v_m és v_x csúcsok ds értékét úgyanúgy változtatja a részgráf változtatása. A függetlenség miatt elég olyan gráfon számolni, amelyre teljesül, hogy $|V_n| = |V_m|$ vagy $|V_n| = = |V_m| - 1$.
Egy megfelelő gráf a P_N, ahol beláttuk, hogy $2 \mid N$ esetén $\max(\delta) - \min(\delta) = \frac{N^2}{4} - \frac{N}{2}$, különben $\max(\delta) - \min(\delta) = \frac{N^2}{4} - \frac{N}{2} + \frac{1}{4}$. □

3.4. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, fa gráfra igaz, ha δ a G ds értékeinek vektora és $|V(G)| = N$, akkor*

$$2(N-1)^2 \leq \sum_i \delta[i] \leq a_N,$$

ahol $a_{n+1} = a_n + n(n+1)$, $a_1 = 0, a_2 = 2$

Bizonyítás. Ismét külön bizonyítjuk az alsó- és felső korlátokat.

Alsó korlát: Az S_{N-1}-ben a $\sum_i (\delta[i]) = 2(N-1)^2$, mert egy csúcshoz $N-1$ darab egy hosszú út tartozik, $N-1$ csúcshoz $N-2$ darab kettő hosszú és 1 darab egy hosszú út tartozik.

$$1 \cdot 1 \cdot (N-1) + (N-1) \cdot ((N-2) \cdot 2 + 1 \cdot 1) =$$

$$(N-1) \cdot (2 \cdot (N-2) + 2) = (N-1) \cdot 2 \cdot (N-1) = 2 \cdot (N-1)^2.$$

A csillag gráfban van a legkisebb átlagos távolság a fa gráfok közül [3]. Az átlagot beszorozva az utak mennyiségével is az előző eredményt kapjuk:

$$\left(2 - \frac{2}{N} \cdot \binom{N}{2}\right) = \left(2 - \frac{2}{N}\right) \cdot N \cdot (N-1) = (2 \cdot N - 2) \cdot (N-1) = 2 \cdot (N-1)^2.$$

Felső korlát:

1. Ha $G = P_1$, akkor nincs út, ami az egyetlen csúcsba vezessen. Az egyetlen csúcsnak a ds értéke 0, a ds értékek összege 0. Ha $G = P_2 = K2$, akkor a

két csúcsban 1-1 egy hosszú út vezet. Mindkét csúcs ds értéke 1. A ds értékek összege 2.

2. Ha $G = P_N$, akkor $\sum_{v,v\in V(P_N)} ds(v) = \sum_{v,v\in V(P_{N-1})} ds(v) + 2 \cdot \sum_{i=1}^{N-1} i$. Mivel a lineáris gráfban (az eggyel kisebb gráfban nem szereplő) új csúcs annyival növeli a többi $N-1$ db csúcs ds értékét, amennyi az ő értéke (hasonlóan az 1. feltételben). Az új csúcshoz vezető utak sorozatot alkotnak, hiszen a gráf lineáris gráf. Ez a sorozat az 1, 2, ... $N-1$. $N-1$ csúcs van az új úton kívül. A $\sum_{i=1}^{N-1} i = \frac{N(N-1)}{2}$, így $\sum_{v,v\in V(P_N)} ds(v) = \sum_{v,v\in V(P_{N-1})} ds(v) + 2 * \frac{N(N-1)}{2}$, amit sorozatként leírva kapjuk az $a_n = a_{n-1} + (n-1\cdot)n$-et, ahol $a_1 = 0, a_2 = 2$.

□

A [17] szerint az útvonal gráfnak van a legnagyobb átlag távolsága adott N esetén. A cikkben szerepelő képletet használva az útvonal gráfra azt kapjuk, hogy az átlag távolsága $\frac{N+1}{3}$. Illetve bizonyítják, hogy összefüggő gráf esetén a legnagyobb átlag távolság $\frac{N+1}{3}$. Minden utat kétszer kell számolnunk, így $2 \cdot \binom{N}{2} \cdot \frac{N+1}{3}$ eredményt kapjuk, hiszen $\binom{N}{2}$ darab út van egy N méretű gráfban. Ez egyezik a sorozat szerinti megközelítésünkkel:

- $n = 1$: $a_1 = 0$, a cikkből származó eredmény: $\frac{0\cdot1\cdot2}{3} = 0$
- $n = k+1$: $a_{k+1} = a_k + k(k+1)$, behelyettesítve a_k helyére a cikkből származó képletet: $(k-1)k(k+1)/3 + 3k(k+1)/3 = [k(k+1)](k-1+3)/3 = = k(k+1)(k+2)/3$.

3.3. Feltételek nem-fa gráfokra

3.5. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, összefüggő, nem-fa gráf esetén, ha $|V| = N$, akkor $2 \mid \sum_{v\in V} ds(v)$.*

A 3.1. Tételben tárgyaltuk már: minden út ugyanannyival növeli a két végcsúcs ds értékét. Az összeg osztható lesz 2-vel.

3.6. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, nem-fa gráf minden v csúcsára igaz, ha $|V(G)| = N$, akkor $N - 1 \leq ds(v) \leq \frac{N(N+1)}{2} - 1$.*

Bizonyítás. Ismét külön bizonyítjuk az alsó- és felső korlátokat.

Alsó korlát: Ha $G = K_N$, akkor a legkisebb ds érték az $N-1$. Ennél közelebb nem kerülhet egyik csúcs sem bármely másikhoz. Nem lehet csökkenteni a $\min(\delta)$-t.

Felső korlát: Fák esetében tudjuk, hogy P_N-ben van a legnagyobb ds érték adott N-re. Egy él hozzáadásával készíthetünk nem fa gráfot. A hozzáadott él legyen (x, y),

$x, y \in V(P_N)$; $(x, y) \notin E(P_N), \deg_{P_N}(x) = 1, d_{P_N}(x, y) = 2$. Így minden csúcs, ami x-től legalább 2 távra helyezkedett el, eggyel közelebb kerül x-hez. Az x ds értéke $N - 2$-vel csökken, ennyi csúcshoz került eggyel közelebb.
Legyen $x' \in V(P_N), \deg(x') = 1, x' \neq x$. A $ds_{P_N}(x') = \frac{N(N+1)}{2}$.
Legyen $G = (V(P_N), \{E(P_N), (x, y)\})$. Ekkor

$$ds_G(x') = \frac{N(N+1)}{2} - 1.$$

A $ds_{P_N}(x')$ a legnagyobb érték a δ_{P_N}-ben. Egy (v_1, v_2) él hozzáadásával (legyen ez a gráf G_2) minden csúcs ds értéke csökken a P_N gráfban, kivéve a v_m csúcsnak ($d_{P_N}(v_1, v_m) = d_{P_N}(v_2, v_m)$, 2.1. Tétel). Mivel $v_m \neq x'$ ($N = 2$-nél nincs ilyen v_m, $N > 2$-nél $ds_{P_N}(v_m) < \max(\delta_{P_N})$) és $ds_{P_N}(v_m) < ds_{P_N}(x')$, így

$$\forall v \in V, v \neq v_m : \; ds_{P_N}(v) > ds_{G_2}(v).$$

Mivel $\max(\delta_G) = ds_G(x') = ds_{P_N}(x') - 1$ és ennél kisebb csökkenést nem lehet elérni (pozitív egész értékkel kell csökkenjen), így

$$\forall v \in V : \; ds_G(x') \geq ds_{G_2}(v).$$

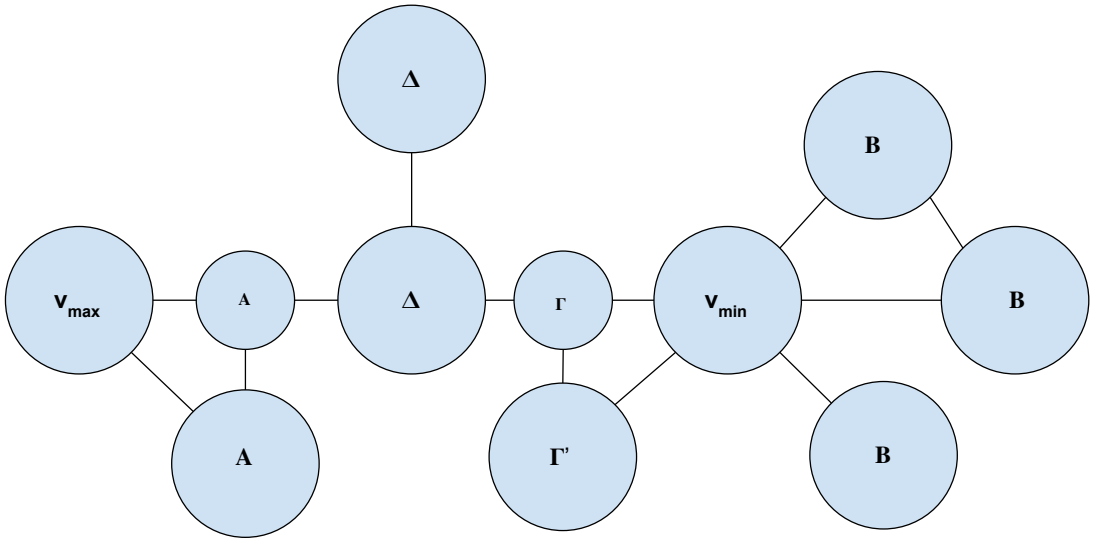
□

3.7. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, összefüggő, nem-fa gráfra igaz, ha δ a G ds értékeinek vektora és $|V(G)| = N$, akkor $2 \mid N$ esetén $\max(\delta) - \min(\delta) \leq \frac{N^2}{4} - \frac{N}{2}$, különben $\max(\delta) - \min(\delta) \leq \frac{N^2}{4} - \frac{N}{2} + \frac{1}{4}$.*

Bizonyítás. Vegyük a $G = (V, E)$ nem-fa gráfot, ahol $|V| = N$.
Legyen $v_{max} \in V, ds_G(v_{max}) = \max(\delta_G)$ és $v_{min} \in V, ds_G(v_{min}) = \min(\delta_G)$, ahol $d_G(v_{max}, v_{min})$ minimális (arra az esetre, ha több csúcs felelne meg, minimális távolságúak közül egy tetszőlegeset választunk ki). A G gráf v_{max} és v_{min} kívüli csúcsait halmazokba sorolhatjuk (példa látható a 3.2. ábrán) :

- Legyen A az a halmaz, amely azokat a csúcsokat tartalmazza, amelyek a v_{max}-hoz vezető útjuk nem tartalmazza v_{min}-t és közelebb helyezkednek el v_{max}-hoz, mint v_{min}-hez.
- Legyen B az a halmaz, amely azokat a csúcsokat tartalmazza, amelyek a v_{max}-hoz vezető útjuk tartalmazza v_{min}-t és távolabb helyezkednek el v_{max}-tól, mint v_{min}-től.
- Legyen Γ az a halmaz, amely azokat a csúcsokat tartalmazza, amelyek a v_{max}-hoz vezető útjuk nem tartalmazza v_{min}-t és távolabb helyezkednek el v_{max}-tól, mint v_{min}-től.

- Legyen Δ az a halmaz, amely azokat a csúcsokat tartalmazza, amelyek egyenlő távolságra helyezkednek el v_{max}-tól és v_{min}-től.



3.2. ábra. *G gráf és a csoportosított csúcsai*

Így $\max(\delta_G)$ a B-beli csúcsok v_{min}-től való távolságainak, $d(v_{max}, v_{min}) \cdot |\mathrm{B}|$, illetve A, Γ, Δ beli csúcsok távolságainak v_{max}-tól és $d_G(v_{max}, v_{min})$ értékek összegéből áll. A $\min(\delta_G)$ pedig az A beli csúcsok v_{max}-tól való távolságainak, $d(v_{max}, v_{min}) \cdot |\mathrm{A}|$, illetve B, Γ, Δ beli csúcsok távolságainak v_{min}-től és $d_G(v_{max}, v_{min})$ értékek összegéből áll. A Δ beli csúcsok távolságai v_{max}-tól és v_{min}-től megegyeznek. Ezáltal

$$\max(\delta_G) - \min(\delta_G) =$$

$$d_G(v_{max}, v_{min}) \cdot |\mathrm{B}| + \sum_{v \in \Gamma} (d_G(v_{max}, v) - d_G(v_{min}, v)) +$$

$$+ \sum_{v \in \mathrm{A}} (d_G(v_{max}, v) - d_G(v_{min}, v)).$$

Tudjuk, hogy $|\mathrm{A}| + |\Gamma| + 1 \leq |\mathrm{B}|$.
A $d_G(v_{max}, v_{min})$-t a továbbiakban jelöljük d-vel. Értékei lehetnek:

- $d = 0$, ekkor $v_{max} = v_{min}$. $\max(\delta_G) - \min(\delta_G) = 0$.
- $d = 1$, ekkor 2.1. Tétel teljesül. $\max(\delta_G) - \min(\delta_G) \leq N - 2 \leq \frac{N^2}{4} - \frac{N}{2}$ vagy $\frac{N^2}{4} - \frac{N}{2} + \frac{1}{4}$ (ha $2 \nmid N$).
- $d = 2$, ekkor $\max(\delta_G) - \min(\delta_G) =$

$$d \cdot |\mathrm{B}| + |\Gamma| - |\mathrm{A}|,$$

mert $\forall v \in$ A-ra $d(v_{max}, v) - d(v_{min}, v) = -1$ és $\forall v \in \Gamma$-ra $d(v_{max}, v) - d(v_{min}, v) = = 1$. $|\mathrm{A}| \geq 0$, így maximális esetben $\max(\delta_G) - \min(\delta_G) = 2 \cdot |\mathrm{B}| + |\Gamma|$, $|\mathrm{B}| + |\Gamma| \leq \leq N - 3$ (mivel $|\Delta| \geq 1$) és $|\mathrm{B}| \leq N - 4$ ($|\Gamma| \geq 1$, különben G egy fa). $N \geq 6$, különben $d \neq 2$. A $2 \cdot |\mathrm{B}| + |\Gamma| \leq 2N - 7 < \frac{N^2}{4} - \frac{N}{2} (+\frac{1}{4})$, minden $N (\geq 6)$ esetén.

- $d > 2$, ekkor $\max(\delta_G) - \min(\delta_G) =$

$$d \cdot |\mathrm{B}| + \sum_{v \in \Gamma} (d_G(v_{max}, v) - d_G(v_{min}, v)) + \sum_{v \in \mathrm{A}} (d_G(v_{max}, v) - d_G(v_{min}, v)).$$

$|\mathrm{A}| \geq 1$ és $\forall v \in$ A-ra $-d < d(v_{max}, v) - d(v_{min}, v) \leq -1$.
$|\Gamma| \geq 1$ és $\forall v \in \Gamma$-ra $1 < d(v_{max}, v) - d(v_{min}, v) \leq d$.
A $d \leq |\mathrm{B}|$, különben $v_{min} \neq \min(\delta_G)$ vagy $v_{max} \neq \max(\delta_G)$ (ami ellentmondás). Ha $d = |\mathrm{B}|$ és $|\mathrm{A}| = |\Gamma|$, B akkor is tartalmazhat kört.
A $d \cdot |\mathrm{B}| + \sum_{v \in \Gamma} (d_G(v_{max}, v) - d_G(v_{min}, v)) + \sum_{v \in \mathrm{A}} (d_G(v_{max}, v) - d_G(v_{min}, v))$ maximum értékének eléréséhez, minden A-beli csúcs a v_{max} és v_{min} közötti legrövidebb úton kell elhelyezkedjen. Csak egy ilyen út létezhet. Így növelve a d-t és nem csökkentve a B és Γ halmazok elemszámát. Mivel a kifejezést közvetetten a Δ halmaz nem befolyásolja, de N fix, így ennek a halmaznak az elemszáma legyen a lehető legkisebb. Ez a d paritásától függően 0 vagy 1. Ha $\Delta = 1$, akkor azt az egy elemet a v_{max} és v_{min} közötti kiválasztott legrövidebb út tartalmazza. A Γ-beli csúcsok közül azok, akik a v_{max} és v_{min} közötti legrövidebb úton helyezkednek el, annyival növelik a kifejezést, mint amennyivel az A-beliek csökkentik. Legyen Γ' a Γ azon részhalmaza, amelynek elemei nem helyezkednek el a v_{max} és v_{min} közötti (kiválasztott) legrövidebb úton. Ekkor $\max(\delta_G) - \min(\delta_G) =$

$$d \cdot |\mathrm{B}| + \sum_{v \in \Gamma'} (d_G(v_{max}, v) - d_G(v_{min}, v)) .$$

$\sum_{v \in \Gamma'} (d_G(v_{max}, v) - d_G(v_{min}, v)) \leq |\Gamma'| \cdot d$, mert Γ'-beli csúcsok legfeljebb csak $d-1$ különbséget produkálnak a v_{max}-tól és a v_{min}-től vett távolságokból (egyenlőség csak is akkor teljesül, ha $|\Gamma'| = 0$). Így

$$d \cdot |\mathrm{B}| + \sum_{v \in \Gamma'} (d_G(v_{max}, v) - d_G(v_{min}, v)) \leq d \cdot |\mathrm{B}| + |\Gamma'| \cdot d.$$

Mivel $|\mathrm{B}| + |\Gamma'| + d + 1 = N$ ($d - 1 = |\mathrm{A}| + |\Gamma \setminus \Gamma'| + |\Delta|$), így

$$\max(\delta_G) - \min(\delta_G) \leq d \cdot (|\mathrm{B}| + N - 1 - |\mathrm{B}| - d) = d \cdot (N - 1 - d).$$

A B halmazbeli csúcsok alkothatnak kört, így G nem fa. A $d = \frac{N-1}{2}$-nél lesz az

érték maximális, ha $2 \nmid N$. Ekkor $|\mathrm{B}| = \frac{N-1}{2}$ és $|\Gamma'| = 0$. Ezáltal

$$\max(\delta_G) - \min(\delta_G) \leq \left(\frac{N-1}{2}\right)^2 = \frac{N^2}{4} - \frac{N}{2} + \frac{1}{4}.$$

Ha $2 \mid N$, akkor $d = \frac{N-2}{2}$ ($d \in \mathbb{Z}^+$) és $|\mathrm{B}| = \frac{N}{2}$. Így

$$\max(\delta_G) - \min(\delta_G) \leq \frac{N-2}{2}\frac{N}{2} = \frac{N^2}{4} - \frac{N}{2}.$$

□

3.8. Tétel. *Minden $G = (V, E)$ egyszerű, irányítatlan, súlyozatlan, összefüggő, nem-fa gráfra igaz, ha δ a G ds értékeinek vektora és $|V(G)| = N$, akkor $N(N-1) \leq \sum_i \delta[i] \leq$ $\leq a_N - 2(N-2)$, ahol $a_{n+1} = a_n + n(n+1)$, $a_1 = 0, a_2 = 2$.*

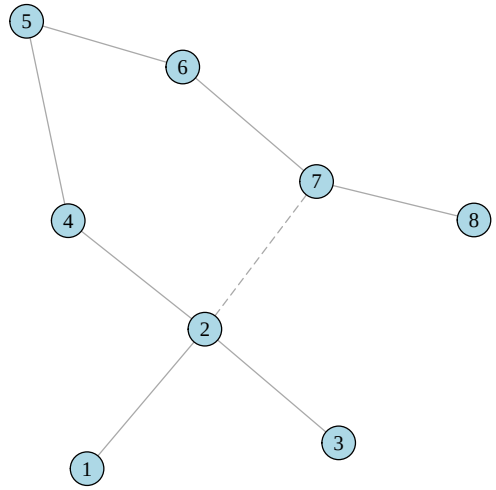
Bizonyítás. Az alsó- és felső korlátokra külön-külön adjuk meg a bizonyítást.

Alsó korlát: Ha $G = K_N$, akkor N db csúcs 1 távolságra helyezkedik el $N-1$ csúcstól. Ennél jobban nem lehet csökkenteni a távolságot bármely 2 csúcs között. Így $N(N-1) \leq \sum_i \delta[i]$.

Felső korlát: Bármely egyszerű gráfhoz úgy adunk új élet, hogy egyszerű gráf maradjon, akkor csökken a csúcsok ds értékeinek összege. Így a legkisebb csökkenést, akkor tudjuk elérni, ha csak egy élet adunk hozzá. Egy fa gráfból egy él hozzáadásával nem fa gráfot kapunk. Fák esetén ismerjük a felső korlátot és létezik olyan fa amelyhez éleket adva adott nem-fa gráfot kapjuk meg, így mondhatjuk, hogy minden nem-fa gráf ds értékeinek összege kisebb egy bizonyos fa ds értékeinek összegétől. Egy felső korlát meghatározásához elegendő azokat az eseteket vizsgálni, mikor a nem-fa gráf egy fához való él hozzáadásával készül.
Legyen $G = (V(G), E(G))$ gráf egy fa, $x, y \in V(G), (x, y) \notin E(G)$.
Legyen $G' = (V(G), \{E(G), (x, y)\})$. Soroljuk be a G'-ben szereplő összes csúcspár közötti legrövidebb utakat, amennyiben egy csúcspárhoz több van, akkor csak az egyiket, ha van olyan amelyik nem tartalmazza (x, y)-t, akkor azt (példa látható a 3.3. ábrán):

- Legyen A azon csúcspárok halmaza, amelyek elemei közötti legrövidebb út hossza G-ben és G'-ben megegyezik.

- Legyen B azon csúcspárok halmaza, amelyek elemei közötti legrövidebb út tartalmazza az x és y csúcsokat G-ben. Így G'-ben a távolság $d_{G'}(x, y) - 1$-gyel kisebb lett.



A :	(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5), (5, 6), (5, 7), (5, 8), (6, 7), (6, 8), (7, 8)
B :	(1, 7), (2, 7), (3, 7), (1, 8), (2, 8), (3, 8)
Γ :	(1, 6), (2, 6), (3, 6), (4, 7), (4, 8)

3.3. ábra. *Egy gráf csúcspárjainak besorolása. Az eredetileg fa gráfhoz a* $(2, 7)$ *élet adnánk hozzá.*

- Legyen Γ azon csúcspárok halmaza, amelyek elemei közötti legrövidebb út tartalmazza x és y csúcsok közül legfeljebb az egyiket és G'-ben kisebb, mint G-ben.

Vizsgáljuk meg a $d = d_{G'}(x, y)$ függvényében a $\sum_{i=1}^{N} \delta_G[i] - \sum_{i=1}^{N} \delta_{G'}[i]$-t.

A $d \leq 3$ esetén

$$\sum_{i=1}^{N} \delta_G[i] - \sum_{i=1}^{N} \delta_{G'}[i] = 2 * (d-1) * |\mathrm{B}| = 2 * (d-1) * (|X| + |Y| - |X \cap Y|),$$

ahol $X = \{v \mid x \in \text{path}(v, y)\}$ és $Y = \{v \mid y \in \text{path}(v, x)\}$. A kettes szorzóval már számoljuk egy út mindkét irányába ható változást, így az X és Y halmazok metszetét ki kell vonnunk. Az $X \cap Y = \{(x, y)\}$, így $|X \cap Y| = 1$. Minden csúcspár (vagy legrövidebb út) A-ba vagy B-ba tartozik, $|\Gamma| = 0$. Az A-beliek nem befolyásolják a különbséget. $|X| + |Y| \geq 2$, mert $x \in X$ és $y \in Y$. Így a csökkenés akkor minimális és $\sum_{i=1}^{N} \delta_{G'}[i]$ maximális, ha $|X| + |Y|$ minimális, vagyis $X = \{x\}$ és $Y = \{y\}$. Ekkor $\deg(x) = \deg(y) = 1$. Látható, hogy $d = 3$ estén a különbség nagyobb, mint $d = 2$-nél, a kérdéses összeg kisebb lesz, így elég vizsgálnunk a $d = 2$ esetet.

Keressük azt a H fa gráfot, amelynek N darab csúcsa van, csúcsai közötti távolságok összege a lehető legnagyobb, x és y csúcsára teljesül, hogy $d_H(x, y) = 2$, x és y csúcsok fokszáma 1.

A H gráf $N - 3$ darab csúcsának elhelyezkedése kérdéses. Legyen H^- a x és y nélküli részgráfja H-nak. Ebben a H^- gráfban a távolságok (és a ds értékek) összege maximális, ha megegyezik P_{N-2}-vel. Már láttuk, hogy adott csúcs mennyiségű gráfok közül az útvonal gráf esetén érjük el a maximális értéket (3.4. Tétel felső korlát bizo-

nyítása). Ekkor x-től és y-tól függ a $\sum_{i=1}^{N} \delta_H[i]$, ami megegyezik az alábbi összeggel:

$$\sum_{i=1}^{N-2} \delta_{H^-}[i] + \sum_{i=1}^{N-2} 2 \cdot (d_H(x, v_i) + d_H(y, v_i)).$$

Mivel $\forall i(1 \leq i \leq N-2) : \ d_H(x, v_i) = d_H(y, v_i)$, így elég azt megvizsgálni, hogy $\sum_{i=1}^{N-2} d_H(x, v_i)$ mikor maximális. Az útvonal gráf csúcsainak ds értékeinek összegét már vizsgáltuk. Ehhez hasonlóan beláthatjuk, hogy egy egytől $N-2$-ig tartó sorozat összegénél lesz maximális az összeg, vagyis a H^- egy leveléhez csatlakozva.

$$H = (\{V(P_{N-1}), y\}, \{E(P_{N-1}), (v_2, y)\}),$$

ahol $(x, v_2) \in E(P_{N-1})$, $x \in V(P_{N-1})$ és $\deg_{P_{N-1}}(x) = 1$.

Így H az a gráf, amelyhez ha hozzáadjuk az (x, y) élet, a lehető legkisebb mértékben csökken a ds értékek összege. Ilyen gráfok közül a legnagyobb $\sum \delta$ értékkel rendelkezik a G'.

Azaz G' megegyezik azzal a gráffal, amit P_N-ből úgy állítunk elő, hogy a lehető legkisebbet csökkenjen a $\sum \delta$, egy levelet (v_N) és ettől a kettő távra lévő csúcsot (v_{N-2}) kötünk össze.

$$(V(H), \{E(H), (x, y)\}) = G' = (V(P_N), \{E(P_N), (v_{N-2}, v_N)\})$$

A P_N gráf ds értékeinek összege $\sum_i \delta_{P_N}[i] = a_N$. Egy (x, y) eddig tárgyalt élet a gráfhoz adva, ahol $\deg(x) = 1$, csökken az x csúcs ds értéke $N-2$-vel, ennyi csúcshoz került közelebb eggyel. A többi $N-2$ darab csúcs, akik nem voltak szomszédosak x-szel, ds értékei eggyel csökkennek. Ezzel a teljes csökkenés $2(N-2)$, így a G' nem-fa gráf ds értékeinek összege:

$$\sum_i \delta_{G'} = a_N - 2(N-2).$$

A $d > 3$ esetén

$$\sum_{i=1}^{N} \delta_G[i] - \sum_{i=1}^{N} \delta_{G'}[i] = 2 \cdot (d-1) \cdot |\mathrm{B}| + 2 \cdot \sum_{p \in \Gamma} d_G(p) - d_{G'}(p).$$

Mivel $\forall p \in \Gamma : \ d_G(p) - d_{G'}(p) \geq 1$, így

$$\sum_{i=1}^{N} \delta_G[i] - \sum_{i=1}^{N} \delta_{G'}[i] \geq 2 \cdot [(d-1) \cdot |\mathrm{B}| + |\Gamma|].$$

A $|\Gamma| = d - 2$, ha $2 \mid d$ és $|\Gamma| = d - 3$, ha $2 \nmid d$. A $d > 3$, így $|\Gamma| > 1$. $|\mathrm{B}| \geq 1$. Ezáltal

$$2 \cdot [(d-1) \cdot |\mathrm{B}| + |\Gamma|] > 2 \cdot (3 \cdot 1 + 1),$$

ami nagyobb, mint $d = 2$ estén. Vagyis bármely H fa gráfra, ha $d > 3$ ($d > 2$), akkor $G'(V(H), \{E(H), (x, y)\})$ nem-fa gráf ds értékeinek összege kisebb, mint $d = 2$ esetén.

Azaz a G' nem-fa gráf csúcsainak ds értékeinek összege nem lehet több az $a_N - -2(N-2)$-nél.

□

3.4. Konklúzió

Bármely N hosszú vektorra sikerült 4 – 4 feltételt megfogalmaznunk, amelyek szükségesek, hogy egy CC értékekből (vagy azok reciprokaiból) gráf rekonstrukciót elvégezhessünk egyszerű, irányítatlan, súlyozatlan fa illetve összefüggő nem-fa gráfhoz. Jövőbeli munkánk további szükséges feltételek megfogalmazása, vagy akár elégségeseké.

4. fejezet

Memetic Differential Evolution hálózattudományi eszközökkel

Ezen fejezet több munkából áll össze. Mindegyik témája a Memetic Differential Evolution algoritmus és a hálózattudomány. Elsőként a központisági mértékekkel változtatott verzióról beszélünk, amely a további munkák alapötleteként szolgált. Majd az alap algoritmus különböző, szakirodalomban ismert változatait vizsgáljuk. Végül az MDE hálózati eszközökkel való fejlesztését tárgyaljuk.

A [31] és [33] cikkekben jelentek meg a fejezetben ismertetett eredmények egy része.

4.1. Bevezetés

Két rendkívül releváns és aktív terület, a globális optimalizálás és a hálózattudomány metszéspontja remek lehetőségeket kínál érdekes kutatómunkákhoz. Ez a fejezet a Memetic Differential Evolution (MDE) algoritmussal végzett kísérletek eredményeit mutatja be, ahol dinamikus segéd gráfokon definiált központisági mértékek vezérelték a végrehajtást.

A Differential Evolution (DE) egy jól ismert iteratív, populáció alapú algoritmus [59]. A memetikus megközelítések minden iterációban lokális optimalizálási módszert használnak, ezért a populáció tagjai mindig a célfüggvény lokális optimumai [49, 51]. Az MDE egy egyszerű kiterjesztése a DE-nek.

Tekintsük a következő globális optimalizálási problémát

$$\min_{\mathbf{x} \in D \subset \mathcal{R}^n} f(\mathbf{x}), \tag{4.1}$$

ahol f egy folytonos függvény, amelyet a Memetic Differential Evolution (MDE) [55] algoritmus segítségével szeretnénk megoldani. A metódus iteratív, populáció alapú és nem analitikus (blackbox) algoritmusok közé tartozik, vagyis csak a függvényértéket

használja az $f(\mathbf{x}) : \mathcal{R}^n \to \mathcal{R}$ alakú függvény globális minimalizálásának megoldásához. A legutóbbi benchmarking eredmények [6, 42] az MDE ígéretes hatékonyságát mutatják a kihívást jelentő optimalizálási problémákkal szemben.

Mint az eredeti DE-ben, minden iterációs lépésben a populáció három elemét kiválasztja egy specifikus formulához, amelyből gyártódik a megoldás-jelölt. Megvizsgáltunk és javasolunk új módszereket a kiválasztási eljáráshoz, amelyeket a hálózattudomány inspirált [52].

4.2. Algoritmusok

4.2.1. Memetic Differential Evolution

A memetikus algoritmusok populáció alapú megközelítések, amelyek lokális keresési módszert tartalmaznak. A populáció minden egyes eleme rendelkezik a probléma egy lokálisan optimális megoldásával (kivéve talán a futtatás legelejét). A Memetic Differential Evolution magas szintű leírása a következő.

1. Kezdjünk egy véletlenszerű populációval: $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\} \quad (\mathbf{p}_i \in \mathcal{R}^n)$.

2. Iterációnként hajtsuk végre a következő lépéseket minden $\mathbf{p}_i$-re, míg el nem érjük a megállási feltételeket:

   a) Válasszunk páronként különböző három elemet a populációból: $\mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l$, mind különbözik $\mathbf{p}_i$-től.

   b) Legyen $\mathbf{c} = \mathbf{p}_j + F \cdot (\mathbf{p}_k - \mathbf{p}_l)$ egy kiválasztott megoldás.

   c) Módosítsuk a $\mathbf{c}$ vektort a CR-keresztezés szerint a $\mathbf{p}_i$ vektorral.

   d) Hajtsunk végre egy lokális keresést a $\mathbf{c}$ vektorból indulva.

   e) Helyettesítsük a $\mathbf{p}_i$ vektort a $\mathbf{c}$ vektorral, ha $f(\mathbf{c}) \leq f(\mathbf{p}_i)$ teljesül.

Az MDE paraméterei: m a populáció mérete, $F \in (0,2)$ a differenciál súly és $CR \in \in (0,1)$ a keresztezés valószínűsége.

A $\mathbf{c} \in \mathcal{R}^n$ megoldás-jelölt CR-keresztezése a 2(c) lépésben a következőképpen hajtódik végre. A megoldás minden koordinátájára egyenletesen generálunk egy r véletlen számot a $(0, 1)$ intervallumból. Ha r nagyobb, mint az adott CR paraméter, akkor a $\mathbf{c}$ adott koordinátájának értékét lecseréljük a $\mathbf{p}_i$ ugyanazon koordinátáján szereplő értékre. Annak érdekében, hogy mindenképpen kapjunk új $\mathbf{c}$ vektort, kiválasztunk és rögzítünk egy koordinátát, amelyet ez a CR-keresztezés átugor (a másik három vektor lineáris kombinációját megtartjuk ezen a koordinátán).

4.2.2. Lokális Optimumok Hálózata (LON)

A Lokális Optimumok Hálózatai, röviden LON-ok (Local Optima Network), érdekes gráfok, melyek optimalizálási problémákkal és gyakorlatilag az optimalizáló eljárásokkal kapcsolatosak. Ebben a munkában egy módosított változatot javasolunk, ahol a csúcsok továbbra is az MDE által talált lokális optimumok, de az élek az MDE iteratív része által generált információkat rögzítik.

Meg kell említsük itt, hogy a [58]-ban más megközelítést dolgoztak ki a DE-vel a LON-ra, ahol a csúcsok a populáció tagjai, és két csúcs akkor van összekötve súlyozott éllel, ha az egy szülője a másiknak (lecserélendő tag - megoldás jelölt). Ennélfogva a generált hálózat a populáció tagjainak fejlődését írja le, nem pedig a lokális optimumok felfedezését.

4.2.3. MDE LON

A következő megközelítést javasoljuk, amely élsúlyozott hálózathoz vezet. A hálózat csúcsai az MDE végrehajtásával kapott megoldások (lokális optimumok). Két csúcs összekötött, ha a megfelelő vektorok szülő-gyerek kapcsolatban vannak (mint a fent említett algoritmus leírás 2(c) lépésében). Ez azt jelenti, hogy minden iteráció végén $\mathbf{c}$ (ami definíció szerint egy lokális optimum) a gráfunk egy csúcsa lesz, és össze lesz kötve $\mathbf{p}_j, \mathbf{p}_k$ és $\mathbf{p}_l$ elemeknek (amelyek szintén lokális optimumok) megfelelő csúcsokkal. Nyilvánvalóan egy lokális optimum megtalálható többször is. Ebben az esetben új csúcsot nem készítünk, de az élek súlya növekedhet. Habár az MDE-ben lévő populáció mérete rögzített, a hálózatunk dinamikusan növekszik az optimalizálási eljárás során.

4.3. MDE LON támogatása

A fent tárgyalt hálózat kiépítése új kiválasztási szabályok létrehozásának lehetőségeit nyithatja meg egy $\mathbf{c}$ megoldás-jelölt elkészítésében, a $\mathbf{c} = \mathbf{p}_j + F \cdot (\mathbf{p}_k - \mathbf{p}_l)$ képletben. Az eredeti DE algoritmusban a három szülő véletlenszerűen egyenletes eloszlással van kiválasztva a populációból. Most, hogy az összes tag hozzá van rendelve az épített hálózat egy csúcsához, használhatjuk a hálózat centralitási mérőszámait.

Általánosságban a centralitás egy gráf csúcsainak egy valós függvénye. A centralitási mértékek számszerűsítik egy csúcs fontosságát, rangsorolhatóvá teszik a csúcsokat. A hálózattudomány hatalmas irodalma különféle definíciókat javasol, amelyek a csúcsok eltérő rangsorolásához vezetnek. A következő mértékeket használtuk a kísérleteink során:

– fokszám centralitás, amely egy csúcs fokszámát használja;

– közöttiség centralitás (betweenness centrality - BC) értéket ad a csúcsoknak abban a mértékben, hogy a csúcsok közötti legrövidebb utak közül hány darabon szerepel;

– PageRank egy csúcsnak olyan értéket ad, amely a hálózatbeli szomszédok értékétől függ, és a kimenő élek száma is befolyásolja;

– közelség centralitás (closeness centrality - CC) az összes többi csúcstól vett távolság összegéből számolódik.

Ezek az eszközök lehetőséget adnak a 2(a) lépésben alkalmazott új kiválasztási szabályok meghatározására. A szülők kiválasztásánál adjunk nagyobb esélyt annak a csúcsnak a kiválasztásra, amely magasabb (vagy kisebb) centralitási mutatóval rendelkezik. A kiválasztás során a tagok valószínűségei arányosak a centralitási értékekkel.

A gyakorlatban ezt a kiválasztási szabályt csak néhány kezdeti iteráció után szabad alkalmazni, hogy az MDE hálózat egyenletes véletlenszerűséggel növekedjen. Bár az általunk épített hálózat mérete növekszik, a kiválasztási szabály csak azokra a csúcsokra vonatkozik, amelyek az aktuális, rögzített méretű populáció tagjai.

4.4. Előzetes numerikus eredmények

Ebben a részben az első prototípus implementációval kapott előzetes eredményeinket mutatjuk be. Ez az AMPL-ben [21] történt, amely elég gazdag nyelv ahhoz, hogy kezelje a vektorhalmazt (mint populációt) és kódolni tudjuk a differential evolution egyszerű ciklusát. Nagy előnye, hogy könnyen használható bármilyen AMPL-kompatibilis megoldó (solver), amely lokális optimalizálóként szolgálhat az MDE 2(d) lépésében. A kérdés továbbra is fennáll, hogyan kell kiszámítani a centralitás mértékét? Véleményünk szerint nem lenne értelme ezeket AMPL-ben implementálni, ezért használtuk a `R/igraph` csomagot. Bár megkönnyítette a kísérletezést, ez a megközelítés nyilvánvalóan nem járható út. Ennek ellenére ez a prototípus azért lett fejlesztve, hogy kimutathassuk a koncepció eredményeit.

A teszteléshez a szakirodalomból a Schwefel tesztfüggvényt választottuk, melynek meghatározása:

$$\mathrm{Schwefel_n}(\mathbf{x}) = 418.9829\mathrm{n} + \sum_{\mathrm{i}=1}^{\mathrm{n}} -\mathrm{x_i}\sin(\sqrt{|\mathrm{x_i}|}) \qquad \mathrm{x_i} \in [-500, 500].$$

Ez a függvény exponenciálisan növekvő számú lokális minimum ponttal rendelkezik, amelyek értéke nagyon közel van a globális optimuméhoz, ráadásul a keresési tartomány különböző régióiban helyezkednek el. Amint arról beszámolnak a [42]-ban,

az MDE meg tudja találni a Schwefel$_n$ globális optimum megoldását viszonylag magas sikeraránnyal a $n = 10, 50$ méreteknél. Ebben a rövid írásban nem törekszünk arra, hogy még magasabb százalékos sikert érjünk el. Ehelyett azt a tényt szeretnénk bemutatni, hogy a javasolt kiválasztási szabály a centralitási értékek alkalmazásával sokrétű megoldáshoz vezethet.

Az MDE-t a különböző kiválasztási szabályok használatával futtattuk $n = 2$ és $n = = 10$ esetén, ahol a populáció mérete $m = 10$ és $m = 20$ volt. A további paraméterek az $F = 0.5$ és $CR = 0.1$ voltak. Az iterációk számát szándékosan 20 lépésen tartottuk az eredmény gráfok átláthatóságáért. Az új kiválasztási szabályok csak az első 10 iterációs lépés után kerültek alkalmazásra.

A 4.1 táblázatban közölt eredmények annak az esetnek felelnek meg, amikor az új kiválasztási szabályokat a populáció összes tagjára alkalmaztuk, a különböző centralitási mutatókkal rangsorolva. A kisebb dimenzió esetén minden verzió legalább egyszer meg tudta találni a globális optimumot. Átlagosan a közelség központiság (CC) teljesített a legjobban. A nagyobb problémára az eredeti DE adta az átlagos és abszolút legjobb megoldást, azonban a globálisat nem találta meg. Az utolsó sorban látható eredményeket úgy értük el, hogy mindig az első három magasan rangsorolt csúcsot választottuk ki. Amint látjuk ez jobb megoldásokhoz vezetett. A CC olyan jó eredményt ért el, mint az eredeti DE.

4.1. táblázat. *Az elért átlagos (és legjobb) függvényértékek. Az utolsó sor azokat az eredményeket mutatja, amikor a 3 leginkább központi csúcsot vettük figyelembe.*

dim	DE	BC	Degree	PageRank	CC
2	26.32 (0)	13.16 (0)	50.44 (0)	52.63 (0)	0 (0)
10	366.62 (118.43)	783.77 (356.83)	748.95 (572.45)	852.03 (690.89)	594.31 (236.87)
10	–	555.75 (236.87)	464.64 (236.87)	503.82 (335.57)	387.26 (118.44)

Nyilvánvalóan nem szabad általános következtetéseket levonni ebben a részben végzett numerikus kísérletekből. Továbbra is kérdés, hogy a hálózatvezérelt kiválasztási szabályokkal rendelkező MDE jobb eredményeket tud-e adni a standard verzióhoz képest. Ennek ellenére továbbra is érdekes megvizsgálni, hogy az optimalizálás végén kapott hálózatok milyen tulajdonságokkal rendelkeznek.

4.4.1. MDE LON-ok tulajdonságai

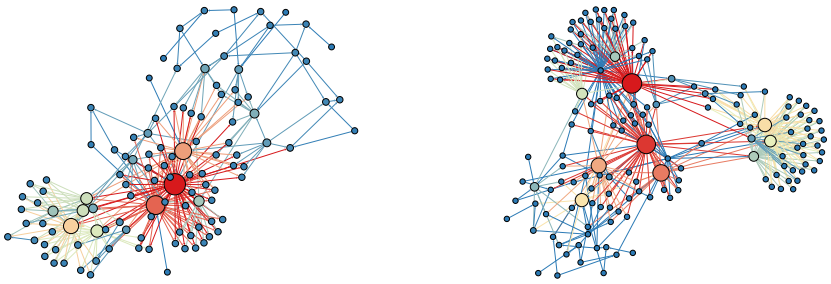
Az MDE algoritmussal épített lokális optimumok hálózataiban szereplő jellemzők vizsgálata sok érdekes eredményt tárhat fel arra vonatkozóan, hogy az optimalizáló hogyan térképezte fel az optimalizálási probléma keresési terét. Itt egy rövid elemzést adunk az MDE végrehajtásával kapott hálózatokról. Néhány alapvető tulajdonság a 4.2 táblázatban található. Az első oszlop a gráf azon tulajdonságait tartalmazza, amelyeket úgy kapunk, hogy nem használunk központi mérőszámot. Láthatjuk, hogy

4.2. táblázat. *A 4.1 táblázat utolsó sorában megadott legjobb megoldásoknak megfelelő MDE LON-ok tulajdonságai.*

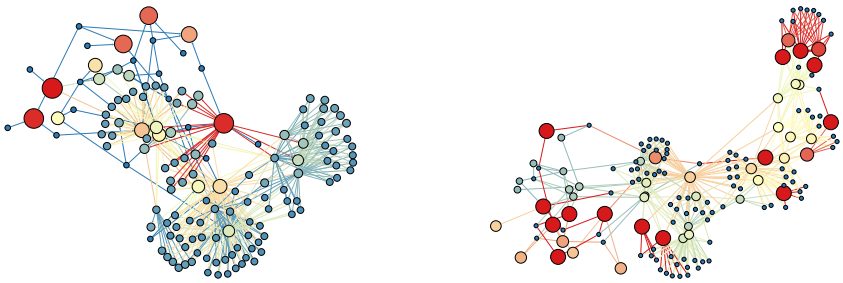
	–	BC	Degree	PageRank	CC
csúcsok száma	179	162	116	145	152
átlagos fokszám	3.11	2.72	2.78	2.76	2.81

a hálózatok az itt jelzett két egyszerű vonatkozásban különböznek egymástól. A csúcsok száma a talált különböző lokális optimumok számát jelenti. Az eredeti MDE tárta fel a legtöbb lokális optimumot. Az új módszerek közül a közöttiség centralitás (BC) találta a legtöbb lokális optimumot, a legjobbat azonban nem tudta megtalálni. Vegyük figyelembe, hogy ha a különböző módszerek ugyanazt vagy eltérő helyi optimumot találtak, ebből a mutatóból nem látható. Az átlagos fokszám egy egyszerű mérőszám lehet, amely sok részletet rejt az eloszlásában, azt jelzi, hogy átlagosan hányszor választottak ki egy csúcsot szülőnek. Itt nem látunk nagy különbséget a centralitás alapú módszerek tekintetében, míg az eredeti kiemelkedik. Az eredmények ismeretében arra a következtetésre jutottunk, hogy ez a hatás előnyös volt az eredeti MDE számára.

A négy hálózat vizualizációja a 4.1 és a 4.2 ábrákon látható. Ne feledjük, hogy a nagyobb csúcsméret nagyobb centralitási értéket jelent, amelyek fokszám, közöttiség, PageRank és közelség értékek. A gráf ábrákon nem szabad szigorú elemzést végezni, mivel a megjelenés az elrendezéstől függ. Ennek ellenére valóban láthatjuk a különbségeket a négy grafikon között, ahogy ezt már a 4.2 táblázat is jelezte. Talán a közelség centralitás alapú hálózat (a 4.2 ábrán jobb oldalon) kiemelkedik, hogy viszonylag sok, magasabb értékű csúcsot tartalmaz. A 4.1 táblázat szerint ez a verzió érte el a legjobb megoldást a hálózatvezérelt MDE-k közül.



4.1. ábra. *MDE LON-ok: fokszám centralitást (balra) és közöttiség centralitást (jobbra) használva. A csúcsméret a megfelelő centralitási értéket jelenti.*



4.2. ábra. *MDE LON-ok: PageRank centralitást (balra) és közelség centralitást (jobbra) használva. A csúcsméret a megfelelő centralitási értéket jelenti.*

4.4.2. Megjegyzések

Ebben a szakaszban ismertetett munka megvalósításakor egy Pythonban készített implementáción dolgoztunk, amely a Pyomo nyílt forráskódú optimalizálási modellezés nyelvét és a centeralitási mérőszámok kiszámításához a `NetworkX` csomagot használja. Ez a környezet lehetővé teszi, hogy mindent a memóriában tartsunk, ami radikálisan lerövidíti a végrehajtási időt a korábban használt prototípus megvalósításhoz képest.

Láttuk, hogy a különböző mérések kis mennyiségű iteráció után is eltérő eredményekhez vezetnek. Ehhez erősen kapcsolódó további vizsgálatokat egy későbbi munkánk során végeztünk, amely a 4.6. fejezetben szerepel.

4.5. Klasszikus DE változatok elemzése

Az előző alfejezetekben bemutatott munkát követően ugyanezen eszközökkel új megközelítésbe kezdtünk. Az új munka bemutatása során először numerikusan megvizsgáljuk a klasszikus $x/y/z$ változatokat az MDE kontextusában. Ezután az MDE algoritmust kibővítjük a lokális optimumok hálózatának (LON) fogalmával. A fejezetben eddig bemutatott, általunk megfogalmazott LON-nal megegyező módon építünk hálózatot. Így az MDE futásának végén egy gráf ábrázolást kapunk arról, hogy a módszer hogyan fedezte fel az optimalizálási probléma terét. A szabványos teljesítménymutatókon kívül a létrejövő LON-ok bizonyos jellemzőit is közöljük és összehasonlítjuk néhány globális mérőszám segítségével. Az egyik részletes elemzés a csúcsok függvényértékei és a kimenő szomszédjaik függvényértékei közötti kapcsolat bemutatása. Ezen és néhány gráftulajdonság alapján javasoljuk az MDE egy a már bemutatottól eltérő kiterjesztését, amely jobb teljesítményt eredményezhet az ebben a szakaszban használt tesztfüggvényeken.

4.5.1. Stratégiák

A legnépszerűbb DE változatokat, amelyek különböző stratégiákat alkalmaznak, a *DE/x/y/z* jelölésekkel különböztetjük meg, ahol

– *x* megadja a perturbálni kívánt megoldást; *rand* vagy *best* lehet, azaz véletlenszerű vagy az aktuális legjobb megoldás.
A fentebb leírt algoritmusban a $\mathbf{p}_j$ kiválasztását határozza meg a 2(a) lépésben.

– *y* meghatározza a számát a különbség vektoroknak (azaz a különbséget két véletlenszerűen kiválasztott populációtag között), amik a perturbációban használunk a 2(b) lépésnél. Tipikus értékek az 1 és a 2.
Az $y = 1$ az alapértelmezett választás, ezért a 2(a) és 2(b) lépés megegyezik a leírásban megadottakkal.
Az $y = 2$ esetén a $\mathbf{p}_k$ és $\mathbf{p}_l$ mellett további két vektor, $\mathbf{p}_m$ és $\mathbf{p}_n$, egy másik differenciálvektor létrehozása érdekében is ki lesz választva.

– *z* határozza meg, hogy melyik valószínűségi eloszlás függvényt használja a keresztezési operátor: *bin* (binomiális) vagy *exp* (exponenciális).
Az *bin*-nél válasszunk véletlenszerűen egy d dimenzió indexet. A 2(c) lépésben a $\mathbf{c}$ vektort változtatjuk meg úgy, hogy minden $e \neq d$ indexre legyen CR valószínűséggel $\mathbf{c}_e := \mathbf{p}_{i_e}$.
Az *exp*-nél válasszunk véletlenszerűen egy d dimenzió indexet. A d-től kezdve lépjünk át minden e dimenzióra és változtassuk $\mathbf{c}_e$-t a $\mathbf{p}_{i_e}$-re. Minden lépésben $1 - CR$ valószínűséggel hajtsuk végre a módosítást.

4.5.2. Hálózat mértékek

Várhatóan a különböző MDE-változatok különböző LON-okhoz vezetnek a futtatásuk végre. Ezen különbségek jellemzésére a teljes lokális optimum gráfon globális mérőszámokat használhatunk, amelyek a következők.

– A csúcsok száma (N) és az élek száma (M);

– az átmérő (D) a legnagyobb érték az összes legrövidebb út hossza közül;

– és az átlagos fokszám (d) (az átlagos bemenő él mennyiség megegyezik az átlagos kimenő él mennyiséggel).

Nagyobb N érték több megtalált lokális optimumot jelent. Az átmérő annak a maximális számnak felel meg, amikor a 2(e) lépés teljesül egy adott populációtag esetében. Végezetül, az átlagos fokszámot tekintve, a $y = 1$ és $y = 2$ variánsoknál a $d < 3.5$ illetve a $d < 5$ a korai konvergenciát jelzi.

4.5.3. A klasszikus változatok tesztelése

Az MDE, a LON készítő és elemző Pythonban lett implementálva a Pyomo [28] és a NetworkX [24] csomagokat felhasználva. A lokális megoldó (solver) az MDE-ben a MINOS [50] volt.

4.5.3.1. Tesztfüggvények

Követve a [6, 42] cikkekben elvégzett numerikus kísérleteket az MDE változatokat két tesztfüggvényen vizsgáltuk:

– Rastrigin:

$$f_R(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), \quad \mathbf{x} \in [-5.12, 5.12]^n,$$

ami egy egytölcséres függvény 10^n darab lokális optimummal, és a globális minimum értéke 0.

– Schwefel:

$$f_S(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right), \quad \mathbf{x} \in [-500, 500]^n,$$

ami többtölcséres függvény 2^n tölcsér aljjal, és a globális minimum értéke $-418.98129n$. A fejezetben eddig bemutatott formától eltér, ahol a globális minimum érték 0 volt.

Valójában ezeknek a függvényeknek a módosított változatait használtuk, nevezetesen a Rastriginen eltolását és forgatását alkalmaztuk: $f_R(\mathbf{W}(\mathbf{x} - \overline{\mathbf{x}}))$, és forgatást a Schwefelen: $f_S(\mathbf{W}(\mathbf{x}))$, ahol a $\mathbf{W}$ egy n-dimenziós ortogonális mátrix, az $\overline{\mathbf{x}}$ pedig egy n-dimenziós eltolási vektor. Ezek az átalakítások még nagyobb kihívást jelentő tesztfüggvényeket eredményeznek, mivel megszűnik két tulajdonságuk, nevezetesen a szétválaszthatóság, valamint az, hogy a globális minimalizálási pontjaik a keresési területük közepén helyezkednek el.

4.5.3.2. Teljesítménymutatók

A tesztfüggvényekhez tartozó eltolási vektorok és forgatási mátrixok rögzítése után $K = 50$ független futást végeztünk minden MDE variációra. A hatékonyságuk összehasonlítására használt teljesítménymutatók a következők.

– S a sikeres futtatások százaléka, azaz hányszor sikerült elérni a globális minimumot;

– 'Best' a legjobb megtalált függvényérték a K futásból;

– 'Avg' a futások átlagos függvényértéke;

– 'Adf' (Average difference failure) az átlagos eltérés a megtalált függvényérték és a globális optimum érték között azokból a futásokból, amelyek nem voltak sikeresek [42];

– 'LS' (Local Search) az átlagos lokális keresések száma sikeres futásonként;

– 'SP' (Success Performance) a siker teljesítmény [6], melyet úgy számítanak, hogy

$$\text{átlag(lokális keresések száma a sikeres futások során)} \times \frac{K}{\text{sikeres futások száma}}.$$

Vegyük figyelembe, hogy minden mutató esetében, kivéve az S-t, az alacsonyabb szám jobb teljesítményt jelez.

4.5.3.3. Megállási feltételek

A következő megállási feltételeket használtuk:

– a jelenlegi populációtagok értékeinek páronkénti különbségeinek összege kisebb, mint 10^{-4};

– a populáció tagjai nem lettek lecserélve az elmúlt 100 iterációban;

– a legjobb megtalált érték nem változott az utolsó 20 000 lokális keresésben (1000 $\times\, m$, ahol m a populáció mérete).

4.5.3.4. Eredmények

Ahogy már említve volt, $K = 50$ független futást végeztünk el minden változathoz. A dimenzió és a populáció méret is 20-ra volt rögzítve. Az MDE paraméterek $F = 0.5$-re és $CR = 0.1$-re voltak állítva minden kísérletben.

A Rastrigin függvény esetében a tesztelt stratégiák mutatói eltérő eredményeket jeleznek, amint az a 4.3 táblázatban látható. A legsikeresebb a *rand/2/bin* változat volt, mivel minden esetben meg tudta találni a globális optimumot. Összességében a *rand/y/z* stratégiák meglehetősen jól teljesítettek, kivéve a *rand/1/exp*-ot, amely a legmagasabb SP értéket eredményezte. A *best/x/y* közül a *best/2/bin* kapta a legmagasabb sikerességi arányt és a legalacsonyabb SP értéket, míg a *best/1/exp* egyáltalán nem ért el sikert. A LON-ok tekintetében észrevehetjük, hogy a *x/2/z* stratégiák a várakozásoknak megfelelően nagyobb grafikonokhoz vezettek. Ez egyértelműen jelzi,

4.3. táblázat. *Mutatók és gráf mértékek a forgatott és eltolt $Rastrigin_{20}$-nál*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	4	0	6.10	6.35	490	12250	358.5	1345.7	10.8	3.74
best/1/exp	0	1.98	10.51	10.5	∞	∞	224.3	837.3	9.24	3.72
best/2/bin	44	0	0.73	1.31	1462.7	**3324**	1370.5	7809.7	12.52	5.69
best/2/exp	14	0	1.48	1.72	1042.8	7449	879.6	5002	12.02	5.68
rand/1/bin	54	0	0.69	1.51	1938.5	3590	1721.8	6954.0	14.38	4.03
rand/1/exp	12	0	2.54	2.88	1393	11611	1106.1	4504.6	14.22	4.07
rand/2/bin	**100**	0	**0**	**0**	6325	6325	6203.7	37212.3	14	5.99
rand/2/exp	92	0	0.09	1.24	3964.3	4309	3817.6	22901.3	13.38	5.99

4.4. táblázat. *Mutatók és gráf mértékek a forgatott $Schwefel_{20}$-nál*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	-7905.9	-7371.6	1007.9	∞	∞	176.1	633.3	7.1	3.57
best/1/exp	0	-8142.7	-7204.9	1174.6	∞	∞	100	341.7	5.9	3.39
best/2/bin	0	-8261.2	-7886.8	492.8	∞	∞	1857.5	10573.4	7.5	5.64
best/2/exp	0	-8024.3	-7629.7	749.9	∞	∞	821.7	4658.6	7.1	5.62
rand/1/bin	2	-8379.6	-7875.2	514.6	3520	176000	2186.8	8676.4	10.8	3.97
rand/1/exp	0	-8024.3	-7639.5	740.1	∞	∞	931.7	3754.9	10.1	4.03
rand/2/bin	**20**	-8379.6	**-8202.2**	**221.7**	15408	**77040**	14946.1	88927.2	9.8	5.94
rand/2/exp	4	-8379.6	-8114.2	276.4	5530	138250	8763.3	52254.4	9.6	5.96

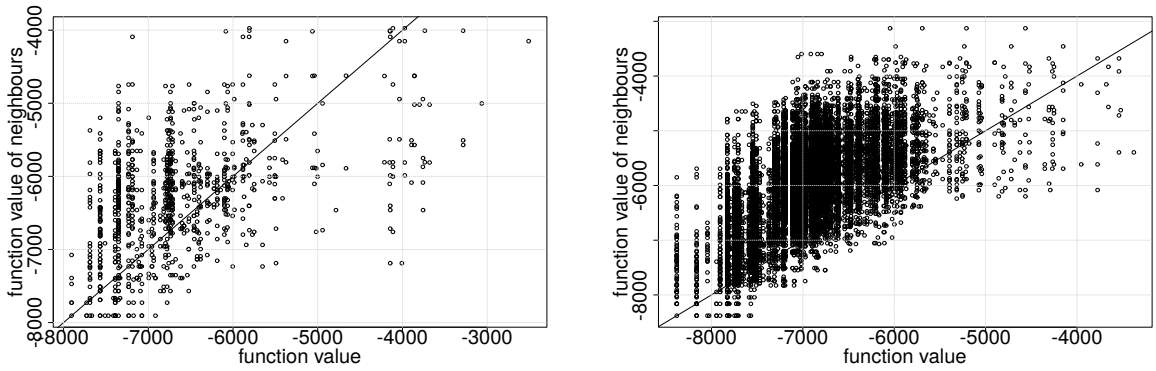
hogy ezek a verziók szélesebb régiókat fedeznek fel az optimalizálás során. Vegyük észre, hogy a nagyobb LON-ok, mint például a *rand/2/z*, nem eredményeztek nagyobb átmérőt. A *best/1/z* stratégiák kis méretű LON-jai és alacsony átlagos mértékük a lokális optimumokhoz való korai konvergencia bizonyítékai.

Ahogy az várható volt, a Schwefel probléma sokkal nagyobb kihívást jelent az MDE verzióknak, lásd a 4.4 táblázatot. A nyolcból csak három stratégia volt képes megtalálni a globális optimumot legalább egyszer. Ennél a függvénynél a *rand/2/bin* rendelkezik a legnagyobb sikeraránnyal és a legalacsonyabb Adf és SP értékekkel, ami lényegében jobb, mint bármely más változat. A *rand/2/bin* viszonylag jó teljesítménye azonban a legtöbb csúcs- és élszámmal rendelkezik a LON-okban, ezért lényegesen több számítási igényű, mint a többi. Általános megfigyelés, hogy az átmérők minden bizonnyal kisebbek a Schwefel problémánál, mint a Rastriginnél. Másrészt az átlagos fokszám értékek nagyon hasonlóak a két problémában.

4.5.4. Hálózatelemzés által támogatott MDE

A LON-ok bemutatásán és az alapvető jellemzőik elemzésén túl, a célunk az MDE algoritmus kibővítése hálózati tulajdonságokat használó szabályokkal, amelyek gazdag információval szolgálnak arról, hogyan hajtódott végre az optimális metódus. Rengeteg lehetőség kínálkozik erre, beszámolunk az egyikről, amely hasznosnak bizonyul

az MDE jobb teljesítmény felé terelésében. Az alábbiakban közölt elemzés alapján javasolhatjuk az MDE módosított változatát.



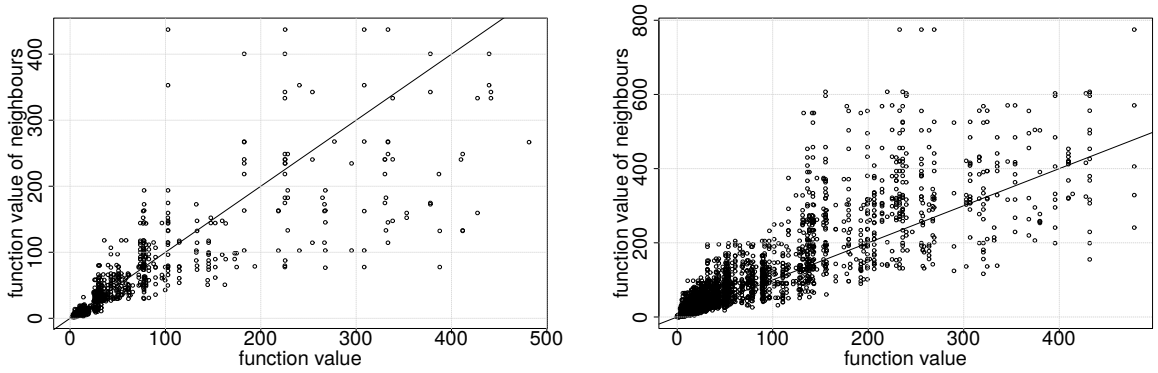
4.3. ábra. *Kimenő szomszédok függvényértékei az f_S-nél $n = 20$-szal; a legsikeresebb futtatások:* best/1/bin *(balra) és* rand/1/bin *(jobbra)*

Az MDE futás során a megfelelő LON felépül, és lehetőség nyílik a csúcsok függvényértékeinek tárolására. Megvizsgálhatjuk az u csúcs kimenő szomszédjait, és összehasonlíthatjuk a függvényértékeiket az u értékével. A 4.3. ábra két ilyen diagramot tartalmaz, amelyek két MDE-változat két különböző futtatását mutatják. Az x tengely a pozitív kimenő fokszámú LON csúcsok függvényértékeit tartalmazza. Minden pont a kimenő szomszédok függvényértékeit mutatja. Az egyenes vonal segít észrevenni, hogy az egyes csúcsokhoz mennyi a magasabb és alacsonyabb függvényértékkel rendelkező szomszédok száma. Ha több pont van a vonal felett, az azt jelzi, hogy az MDE-változat több, rosszabb függvényértékű gyermeket hozott létre egy adott csúcsból. Ennek a viselkedésnek a mellékhatása a keresési tér szélesebb körű felfedezése, ami különösen előnyös lehet a többtölcséres függvényeknél, mint például a Schwefel.

Bár a *rand/1/bin* változat sokkal nagyobb LON-okat eredményezett, mint a *best/1/bin* változat, a 4.3. ábra világosan mutatja, hogy a *rand/1/bin* viszonylag sokkal több ponttal rendelkezik a vonal *felett*, mint *alatta*. A többi *rand/y/z* változat esetében is hasonló ábrákat kaptunk, és a 4.4. táblázatból tudjuk, hogy ezek közül néhány változat képes volt megtalálni a globális minimumot. Másrészt a *best/1/bin* elakadt egy lokális minimum pontban, és az ábrából láthatjuk a mohó viselkedés jelét.

A 4.4. ábrán látható, hogy a sikeresebb változatok hasonló viselkedést mutatnak az egytölcséres Rastrigin függvénynél. A függvényhez való mohó viselkedés jobb teljesítményt eredményezhet, ennek ellenére a *best/1/exp* legsikeresebb futtatása (az elért legjobb függvényérték szempontjából) is konvergált egy lokális minimumhoz (a 4.4. ábra bal oldala).

Ezen megfigyelések alapján készek vagyunk javasolni az MDE kiterjesztését LON használatával.



4.4. ábra. *Kimenő szomszédok függvényértékei az f_R-nél $n = 20$-szal; a legsikeresebb futtatások:* best/1/exp *(balra) és* rand/1/exp *(jobbra)*

4.5.4.1. Above-Below szabály

A korai konvergencia elkerülése érdekében javasolunk egy a populáció néhány tagját érintő újraindítási szabályt. Csak azon tagok a problémák, amelyek a konvergenciát generálták, amíg az MDE nem fedezte fel a keresési tér elég nagy részét, szóval azok a csúcsok, amelyeknek több szomszédja van a vonal alatt. Távolítsuk el ezeket az egyedeket a populációból, és vegyünk be véletlenszerűen újakat. Az állandó újraindítás megakadályozná a konvergenciát, ezért az újraindításra csak az MDE minden α-dik iterációjában kerülhet sor. Észrevettük, hogy amikor a LON átmérője elég nagy, akkor a populáció a tér meglehetősen nagy részét látogatta meg, így jó eséllyel konvergál a globális optimumhoz, ha az MDE-t e módosítás nélkül használjuk.

Javasoljuk az MDE algoritmus kiterjesztését a 2. lépésében a következő szabállyal, amelynek három egész paramétere van: $\delta > 0, \alpha > 0$ és $\theta \leq 0$. Ha az aktuális LON átmérője kisebb mint δ, akkor minden α-dik iterációban az összes $\mathbf{p}_i$-re csináljuk a következőket:

– gyűjtsük össze a $\mathbf{p}_i$ kimenő szomszédait N_i halmazba,

– számítsuk ki az $N - i$ elemeinek függvényértékét,

– legyen $N_i^a := \{\mathbf{q} \ : \ f(\mathbf{q}) > f(\mathbf{p}_i)\}$, és $N_i^b := \{\mathbf{q} \ : \ f(\mathbf{q}) < f(\mathbf{p}_i)\}$

– ha $|N_i^a| - |N_i^b| < \theta$, akkor $\mathbf{p}_i$-t cseréljük le egy véletlenszerűen gyártott vektorra.

Megjegyezzük, hogy a csúcsok függvényértéke a LON-ban tárolva van, így praktikusan csak egyszer kell kiszámolni.

4.5.4.2. Numerikus kísérletek

Használva a fentebb bemutatott szabállyal kiterjedt numerikus vizsgálatot végeztünk, hogy lássuk a teljesítménymutatókat. Célunk az volt, hogy megtaláljuk a három paraméter kombinációját, amely hatékonyságnövekedéshez vezet. Ezért paraméterenkénti kijelölt értékek összes kombinációját kipróbáltuk: $\delta \in [6, 9]$, $\alpha \in [3, 6]$, és $\theta \in [-2, 0]$. A δ értékeinek az intervallum kiválasztását az indokolja, hogy a 4.4. és 4.3. táblázatok szerint a LON-ok átmérői adott MDE változathoz sokkal nagyobb a Rastriginnél, mint a Schwefelnél, és soha nem lépi túl a 10-et az f_S esetén. Másrészt, amikor az f_R-nél a populáció tagjainak már 0-hoz közeli függvényértékei vannak, akkor nem bölcs dolog az MDE-t a keresési tér feltárására kényszeríteni.

Csak az $n = 20$ paraméterű kísérletek eredményeit mutatjuk be. Eredményeink szerint a $\delta = 7, \alpha = 3, \theta = -1$ kombináció eredményezte a legjobb teljesítménynövekedést a tesztelt függvények esetében. A mutatókat a 4.5. és 4.6. táblázat tartalmazza, ahol a javított mértékeket aláhúzással emeljük ki.

4.5. táblázat. *Mutatók és gráf mértékek a forgatott és eltolt $Rastrigin_{20}$-nál az Above-Below újraindítási feltétellel*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	0.99	5.56	5.56	∞	∞	378.1	1424.3	11.52	3.76
best/1/exp	0	1.99	17.38	17.38	∞	∞	226.4	843.5	9.52	3.71
best/2/bin	60	0	0.61	1.54	1449.3	**2415**	1349.8	7677.1	12.88	5.68
best/2/exp	18	0	2.39	2.92	993.3	5519	879.8	4989.7	12.16	5.66
rand/1/bin	60	0	0.55	1.39	1996.0	3327	1794.4	7244.1	14.96	4.03
rand/1/exp	14	0	2.21	2.57	1411.4	10082	1110.1	4521.1	14.1	4.07
rand/2/bin	**100**	0	**0**	**0**	6341.6	6342	6225.1	37318.3	14.64	5.99
rand/2/exp	98	0	0.03	1.78	3897.1	3977	3776.1	22641.9	14.04	5.99

Láthatjuk, hogy az újraindítási szabályunk akár 16%-ot is javított az egytölcséres Rastrigin függvény sikerességi százalékán (S), és nyolc változatból hat esetében alacsonyabb átlagos függvényértéket eredményezett. A *best/2/bin*-nél 7% javulást értünk el a sikerteljesítmény terén.

A többcsatornás Schwefel függvény esetében nem segít megtalálni a globális otimumot az új szabály azoknak a változatoknak, amelyek sikertelenek voltak az eredeti verzióban. Ez azonban átlagosan alacsonyabb függvényértékű lokális optimumok megtalálására késztette őket, és ezáltal csökkentették az „átlagos eltérési hiba” (Adf) mértékét. A leghatékonyabb változat, a *rand/2/bin*, jobb lett az SP mértékben 32%-kal.

4.6. táblázat. *Mutatók és gráf mértékek a forgatott $Schwefel_{20}$-nál az Above-Below újraindítási feltétellel*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d
best/1/bin	0	-8142.7	-7685.6	693.9	∞	∞	278.1	1014.3	7.3	3.63
best/1/exp	0	-8024.3	-7464.8	914.8	∞	∞	149.8	532.8	6.3	3.51
best/2/bin	0	-8261.2	-7993.3	386.2	∞	∞	1991.5	11323.6	7.4	5.66
best/2/exp	0	-8261.2	-7834.0	545.6	∞	∞	1731.6	9869.2	7.5	5.67
rand/1/bin	0	-8142.7	-7899.7	479.8	∞	∞	2370.9	9408.0	11.2	3.97
rand/1/exp	0	-8261.2	-7639.2	740.4	∞	∞	1065.6	4270.5	10.2	4.01
rand/2/bin	**26**	-8379.6	**-8188.6**	**258.1**	13524	**52017**	16304.1	96986.8	9.9	5.94
rand/2/exp	4	-8379.6	-8111.9	278.8	5850	146250	8384.2	50048.2	9.8	5.96

4.7. táblázat. *Mutatók, gráf mértékek és átlagos futási idő $Rastrigin_{10}$-nél hálózati szabályokkal*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d	AvgTime
CC	49	0.0	0.02	0.995	2576.94	2598.50	757.94	3679.0	11.02	4.85	283.4
BC	48	0.0	0.04	0.995	3263.13	3311.41	799.62	3568.42	11.36	4.46	485.9
PR	**50**	0.0	**0.0**	**0**	2679.0	2679.0	755.04	3650.08	10.6	4.84	157.4
Deg	**50**	0.0	**0.0**	**0**	2797.2	2797.2	775.84	3636.22	11.08	4.69	37.23
HC	49	0.0	0.04	1.99	2466.73	**2491.67**	760.04	3678.64	10.84	4.84	235.12
Uni	**50**	0.0	**0.0**	**0**	2499.4	2499.4	739.9	3580.9	10.12	4.84	121.58

4.6. Centralitási mértékeken alapuló szabályok

Az MDE közismert verziónak hálózat alapú vizsgálatát követően úgy gondoltuk, hogy a szülők kiválasztását is az épülő hálózat határozhatná meg. Az *x/1/bin* verzióhoz hasonlóan készülne az utód, de a három (vagy más verzióknál öt) populáció tagot a centralitási értékek szerint választanánk ki. Ezáltal a populáció tagjait rangsorolhatjuk. A lecserélendő tagot kihagyva a három legnagyobb (vagy legkisebb) értékű tagot választjuk. A fejezet elején bemutatott módhoz (4.3. alfejezet) hasonlóan, de nem valószínűség alapon.

Numerikus teszteket végeztünk a már ismertetett Rastrigin és Schwefel függvényeken. A Schwefel függvény azon formáját teszteltük, ahogy a 4.4. alfejezetben is írtuk, az ismert globális optimum értéket hozzáadtuk, így a globális optimum értéke 0. A populáció mérete mindig $m = 10$ volt, de a függvények dimenziója szerint $n = 10$ és $n = 20$ eseteket vizsgáltuk. A használt centralitási mértékek a közelség centralitás (CC), közöttiség központiság (BC), PageRank (PR), fokszám (Deg), harmonikus centralitás (HC) volt. Az összehasonlíthatóságért a tesztek során az egyenletes véletlen kiválasztási szabályt is végrehajtottuk (Uni), ami a *rand/1/bin* változat. Minden szabály-probléma pároshoz 50 futást hajtottunk végre.

A 4.7.-4.10. táblázatok alapján az egyik legjobban teljesítőnek az eredeti Uniform Dist (*rand/1/bin*) változat bizonyult. Bár a többi szabály hasonlóan teljesített, az

4.8. táblázat. *Mutatók, gráf mértékek és átlagos futási idő $Rastrigin_{20}$-nál hálózati szabályokkal*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d	AvgTime
CC	45	0.0	0.14	1.399	8996.0	9151.36	1977.48	9383.7	16.86	4.75	3718.33
BC	39	0.0	0.24	1.085	10570.26	11026.96	2135.92	9394.0	16.32	4.4	6696.98
PR	**49**	0.0	**0.02**	**0.995**	8360.61	**8437.11**	1996.14	9454.72	16.6	4.74	1106.4
Deg	**49**	0.0	**0.02**	**0.995**	8540.61	8584.76	2117.1	9712.76	16.04	4.59	956.2
HC	48	0.0	0.04	**0.995**	8553.75	8697.049	1977.46	9356.34	16.48	4.74	2218.81
Uni	**49**	0.0	**0.02**	**0.995**	8963.67	8972.51	2017.54	9552.84	16.96	4.74	1024.23

4.9. táblázat. *Mutatók, gráf mértékek és átlagos futási idő $Schwefel_{10}$-nél hálózati szabályokkal*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d	AvgTime
CC	17	0.0	108.57	164.5	13440.0	14908.3	2744.58	12236.22	7.52	4.45	1477.62
BC	12	0.0	125.58	165.23	18045.83	19906.25	2478.8	10365.28	9.82	4.15	3361.19
PR	17	0.0	108.57	164.5	13481.18	14205.88	2744.14	12285.56	7.72	4.46	495.62
Deg	11	0.0	125.15	**160.45**	22807.27	22090.9	3029.98	13089.98	9.4	4.31	431.12
HC	14	0.0	125.15	173.82	16694.29	17272.96	2831.1	12677.52	8.32	4.47	1388.99
Uni	**20**	0.0	**105.02**	175.03	11954.5	**11883.75**	2893.72	12949.2	7.68	4.47	397.19

4.10. táblázat. *Mutatók, gráf mértékek és átlagos futási idő $Schwefel_{20}$-nál hálózati szabályokkal*

szabály	S	Best	Avg	Adf	LS	SP	N	M	D	d	AvgTime
CC	0	118.44	598.27	598.27	∞	∞	7297.08	30467.04	10.6	4.18	6386.96
BC	0	118.44	613.63	613.63	∞	∞	7268.5	29769.94	12.76	4.1	32271.98
PR	0	118.44	525.56	525.56	∞	∞	7372.44	30788.12	10.48	4.18	1410.93
Deg	**1**	0.0	542.57	553.64	444270	**540500**	7413.58	30591.98	11.88	4.13	1130.82
HC	0	118.44	560.82	560.82	∞	∞	7733.28	32283.02	10.26	4.18	4723.68
Uni	0	118.44	**505.79**	**505.79**	∞	∞	7353.1	30723.8	10.6	4.18	1130.77

extra műveletek elvégzése miatt ebben a formában nem ajánljuk ezeket a változtatásokat.

4.7. Konklúzió

A fejezetben bemutatott első munka (4.3.-4.4. alfejezetek) során az MDE futása során keletkező hálózatot és a hálózat által vezérelt valószínűségen alapuló kiválasztási szabályok kezdeti eredményeit mutattuk be.

Legjobb tudomásunk szerint a 4.5.3.-4.5.4.2. alfejezetekben szereplő munkánk az első, amely az MDE-változatok benchmarking eredményeiről számol be. A numerikus kísérletek szerint a *rand/2/bin* stratégia összességében a legjobb százalékos sikermutatót biztosította, különösen, ha többtölcséres problémára alkalmaztuk. Ez némileg összhangban van a [47]-ben közölt eredményekkel DE esetében. Egytölcséres függvények esetén a *best/2/bin* változat előnyös lehet, ha jó sikerteljesítményre, azaz alacsonyabb számítási időre van szükség.

Megmutattuk, hogy az MDE lokális optima hálózatára vonatkozó bizonyos ismeretek beépítése az evolúciós folyamatba elvezethet ahhoz, hogy formalizáljuk egy újraindítási szabályt a populáció diverzifikációjának fokozása érdekében. Numerikus tesztjeink azt mutatják, hogy a javasolt újraindítási szabály átlagosan előnyös az MDE változatok többségénél.

Ebben a munkában olyan számítási eszközt fejlesztettünk ki Pythonban a Pyomo és a NetworkX csomagokat használva, amely általános keretet ad számunkra, hogy további lehetőségeket fedezzünk fel az (evolúciós) globális optimalizálás és hálózattudomány területén.

Az utolsó alfejezet azon munkát tárgyalja, amiben a kiválasztási szabályok hasonlítanak a fejezet elején találhatókhoz. Az elvégzett tesztek és vizsgálatok eszközei a második munkában bemutatottakkal egyeznek meg. Pozitív eredményt jelenleg nem sikerült elérnünk itt. További módosítások és tesztelések szükségesek.

5. fejezet

Befolyás terjedés maximalizálás probléma terének vizsgálata

A fejezetben a befolyás terjedés maximalizálás problémát vizsgáltuk a lokális optimumok hálózatával. A probléma és az eszköz is gráfon alapszik. Az adott probléma és algoritmus párosból speciális gráfot készítünk, mely a probléma egy bizonyos vizualizálására alkalmas. A készített gráffal a probléma megértését és új, hatékony eljárások megalkotását szeretnénk elősegíteni.

A fejezetben ismertetett munka a [32] cikkben jelent meg.

5.1. Bevezetés

Legyen adott egy irányított, súlyozott hálózat, továbbá egy terjedési modell és egy k egész szám. Feladatunk, hogy kiválasszunk azt a k darab csúcsot, amelyekből a terjedési modellt indítva és végrehajtva a lehető legtöbb csúcsot elérjük. Az így kapott probléma, a befolyás terjedés maximalizálás, gyakran használt eszköz például a vírusmarketingben. Gazdasági szempontból egy példa lehet a következő: korlátozott mennyiségben van reklámozásra költségünk, maximalizálni szeretnénk az információ terjedést a termékünkről, tehát a lehető legtöbb emberhez szeretnénk eljuttatni. A gráf reprezentálja a közösséget, melyben terjesztenénk az információt. Mint szociális hálózat, nem tartalmaz többszörös éleket és hurkokat, valamint minden egyed különböző mértékben tud hatni a többire.

A befolyás terjedés hálózatokon, mint diszkrét optimalizálási feladat először Kempe és szerzőtársai [37] cikkében jelent meg, amely Domingos és Richardson korai munkájára [16] alapszik. Ebben a mostanra klasszikussá vált cikkben adták meg a szerzők a fejezetben is használt független kaszkád (independent cascade, IC) valamint a lineáris küszöb (linear threshold, LT) modelleket. Továbbá, ebben a cikkben javasolták a mohó algoritmust, amely a feladat matematikai tulajdonságából adódóan legalább $1 - 1/e$ közelítés ad az optimális értékre. Megmutatták, hogy a probléma

az alap terjedési modellekben (IC és LT), melyeket figyelembe vettek, NP-nehéz.

Amennyiben a megadott modellek naiv implementációjával kísérletezünk, hamar rájöhetünk, hogy már akár százas nagyságrendű csúcs-számmal rendelkező gráfra is nagyon sokáig tart lefuttatni a mohó algoritmust (vagy szinte bármilyen más optimalizáló eljárást). Ennek egyik oka, hogy a terjedési modellek sztochasztikus alaptulajdonságát többször egymás utáni lefuttatással tudjuk kiátlagolni (azaz empirikus várható érték közelítését számolni). Ezért hamar megjelentek, és azóta is az ide vonatkozó kutatások egyik fókuszába tartoznak azok a cikkek, amelyek a feladat gyorsabb megoldását tűzték ki célul. [41] az LT modellre ad egy ú.n. lazy-forward mohó algoritmust CELF néven, amelyet aztán a [23] cikkben fejlesztenek tovább CELF++ néven, amelyek az eredeti mohó eljárás iterációinak számát csökkentik nagyságrendekkel. További gyorsítási módszereket is találunk a szakirodalomban, például olyat, amely közelítő eljárást javasol [40], vagy pedig valamilyen strukturális tulajdonságot, például közösségek jelenlétével hozza összefüggésbe a leghatékonyabban aktivizáló csúcsok kiválasztását [56].

Az alapfeladat helyett, ahol tehát egy megoldást akkor tekintünk jobbnak, ha minél több befolyás alá vont csúcsot eredményez, másképpen is megfogalmazhatunk ide kapcsolódó optimalizálási problémákat, vagy változtathatunk az alapgráf tulajdonságain. Ilyen lehet például az, amikor nem csak pozitív, hanem negatív súlyokat is megengedünk. Különösen érdekes a dinamikusan változó gráfok esete is [1]. Ebben a munkában, csak az eredeti alapfeladattal foglalkozunk.

Célunk határozottan nem az, hogy hatékony algoritmust találjunk a feladat megoldására, hanem az, hogy a feladat szerkezeti tulajdonságait mélyebben megértsük. Ehhez felhasználjuk a lokális optimumok hálózatának (local optima network, LON) koncepcióját. A módszerrel tulajdonképpen az eredeti hálózatból, mint a keresési tér értelmezési tartományából, egy speciális, hegymászó-jellegű optimalizáló eljárás felhasználásával készítünk egy másik hálózatot, a LON-t, amelynek segítségével a keresési tér értékkészletének különböző strukturális tulajdonságait vizsgálhatjuk.

5.2. Definíciók és jelölések

Ebben a szakaszban leírjuk a későbbiekben használt legfontosabb definíciókat, jelöléseket. A feladatban használt súlyozott, irányított gráfot $G(V, E, W)$-vel jelöljük.A csúcsoknak azt a k elemű részhalmazát, amelyet a feladat szerint meg kell keresnünk, az angol nyelvű szakirodalmat követve, *seed halmaznak* nevezzük. A gráfnak azon csúcsait, amelyeket a befolyás terjedési modell elér, *aktív* vagy *fertőzött* csúcsoknak nevezzük. A továbbiakban először bemutatjuk a munkánkban használt terjedési modellt, röviden leírjuk a [37] által javasolt mohó algoritmust, majd bevezetjük a LON építésre használt hegymászó-jellegű algoritmust.

5.2.1. Befolyás terjedési modell

Mint említettük, a [37] cikkben két terjedési modellt vezettek be és vizsgáltak, amelyekből jelen munkában csak az egyiket használjuk: a *független kaszkád* (independent cascade, IC) modellt. Ebben az iteráció alapú modellben a gráf minden élének van egy paramétere p ($0 \leq p \leq 1$), amely az élen való terjedés valószínűségét jelzi. Egy iterációban a frissen fertőzött csúcsok megpróbálják aktiválni a (ki)szomszédaikat az él paramétere szerint. Amennyiben egy v csúcs nem tudta fertőzni w szomszédját először, akkor nem fogja tudni később sem, csak egyszer próbálkozhat. Ha egyik aktív csúcs sem tud új csúcsot megfertőzni, akkor az iteráció véget ér. Mivel az IC (és hasonlóan egyébként az LT) modell tartalmaz sztochasztikus paramétert, amely az ismeretlen hatásokat testesíti meg, így a kiértékeléséhez szükséges R-szer megismételni a szimulációt, majd a kapott értékek átlagát venni, vagyis Monte Carlo-módszerrel kiértékelni.

Az időbeli hatékonyságért az IC modellt nem az eredetileg megfogalmazott módon számítottuk, hanem a [25] cikkben javasolt változatot követve készítettünk egy implementációt. Készítünk R darab másolatot az eredeti G gráfról. Az él súlyait valószínűségekként alkalmazzuk. Másolatonként minden csúcsból fertőzést szimulálunk a szomszédokba. Ha a fertőzés egy másolatban nem tudott az élen átjutni, akkor az élet abban a gráfban töröljük. Ha ki akarunk értékelni egy seed halmazt, akkor csak gráfkereső algoritmusokat (mélységi vagy szélességi keresőt) kell indítanunk a seed halmaz minden csúcsából minden másolat gráfban, majd összeszámolni (multiplicitás nélkül) hány különböző csúcsot értek el egy gráfon belül és venni az átlagát ezen értékeknek. Ez tehát a várható (befolyás) értéke a seed halmaznak. Az R határozza meg a pontosságát ennek a várható értéknek. Ezen másolat gráfokat megtarthatjuk a többi kiértékeléshez is, nem kell újakat generálni.

5.2.2. Mohó algoritmus

Mivel a befolyás terjedés maximalizálás feladata az IC terjedési modell mellett szubmoduláris [37], ezért egy mohó algoritmus az optimális megoldás approximációját $1 - 1/e$ faktorral megadja. Ebben a kontextusban az algoritmus kezdetben készít egy S üres halmazt, beállítja a $k = 1$-et és megoldja a problémát. A legjobb értékű csúcsot, a megoldást, behelyezi az S-be. Ezután növeli a seed halmaz lehetséges méretet eggyel és kiértékeli az összes $S \cup v$ ($v \in V \setminus S$) seed halmazt. A legjobb értékhez tartozó v csúcsot S-be helyezi. Addig ismétlődik ez, míg S mérete el nem éri az eredeti problémában megadott k elemszámot.

5.2.3. Egy hegymászó-jellegű algoritmus

Az imént tárgyalt mohó algoritmusban nem definiálható a lokális optimum fogalma. Ezért bevezetünk egy –az eredeti feladat megoldására alkalmas, de jelen munkában nem arra használt– optimalizáló eljárást, amely jellegéből adódóan egy hegymászó algoritmus.

Az eljárás első lépésként egy adott (véletlenszerűen vagy más módon választott) S seed halmazból kezdi a keresést. A második lépésben kiértékeli az aktuális seed halmazt, választ véletlenszerűen az S halmazból egy csúcsot és azt leváltja az egyik (be)szomszédjára, amely jelenleg nem eleme S-nek. Ha az így kapott új $\hat{S}$ seed halmaz értéke jobb, mint az előző, akkor megtartjuk, azaz legyen $S := \hat{S}$ és ismételjük a második lépést. Ha nem, akkor visszatérünk a S seed halmazra és keresünk egy új, megfelelő szomszéd csúcsot, amellyel elvégezzük a kiértékelést és lehetséges cserét ugyanúgy, mint az előbb. Amennyiben a seed minden lehetséges szomszédjaiból alkotott kombinációt bejártuk és egyik sem rendelkezik nagyobb befolyásértékkel, akkor a seed halmazt *lokális optimumnak* nevezzük. Ebben az esetben a hegymászó eljárásunk véletlenszerűen választ egy elemet a seed halmazból és lecseréli egy csúcsra (nem feltétlen szomszédra) a gráfból, amely nem szerepel benne. Ezen csere után az elejétől ismételjük az algoritmust.

Minden kiértékelést (seed elemeket és befolyásértéket) bejegyzünk egy táblázatba a hozzátartozó lokális optimummal (utólagosan), amelybe az algoritmus eljutott a szomszédokra való lépésekkel. Amennyiben egy már látott seed halmazra lép (ezt a táblázatból tudjuk), akkor nem számoljuk ki újra a befolyásértéket, csak kikeressük a hozzá tartozó lokális optimumot. Amennyiben nincs hozzá bejegyezve, mert rosszabb értékkel rendelkezett és nem rajta keresztül lépett tovább, illetve amennyiben kört alkotva visszajutott, akkor csak a befolyásértéket adjuk vissza. A befolyásérték számítása determinisztikus, mivel változatlanul a rögzített R darab másolat gráfon hajtódna végre. Az eljárás ilyen módon történő determinizálása csökkenti a számításigényt. Minden látogatott seed halmazt legfeljebb egy lokális optimumhoz rendelünk, mely a szomszédságban lévő seed halmazok befolyásértékétől függ.

Mint azt említettük, tehát az egyik fő különbség a mohó eljárás és az imént definiált hegymászó-jellegű módszer között, hogy az utóbbival definiálhatunk lokálisan optimális megoldásokat a problémában. Ez a lehetőség vezet minket a következő alfejezethez, amiben tárgyaljuk hogyan lehet hálózatot építeni ezen lokális optimumokból.

5.3. Lokális Optimumok Hálózata - LON

Az imént bevezetett hegymászó-jellegű algoritmus végrehajtása során kaphatunk olyan csúcs halmazokat, amelyek lokális optimumként értelmezhetőek. Ezen lokális opti-

mumokból készítünk egy gráfot, egy LON-t, aminek fogalmát már tárgyaltuk korábban. A LON közelítéseket vizsgálták folytonos [63] és kombinatorikus optimalizálási problémákban [14]. A befolyás terjedés maximalizálás az utóbbihoz tartozik, másként definiáljuk a LON-t ebben a fejezetben. Minden, a hegymászó által megtalált lokális optimum egy csúcs a LON-ban. Két csúcs között él helyezkedik el, ha a kereső algoritmus egymás után talált rájuk: egy lokális optimumból kilépve (egy elemet lecserélve) eljutott a másik lokális optimumba. Ez egyben irányítást is ad a LON-ban. Mivel a hegymászó képes a LON már meglévő élein többször is áthaladni az építés alatt, ezért ebben a munkában a LON súlyozott gráf.

5.3.1. Megállási feltételek

A LON gráf építése tulajdonképpen tetszőleges ideig tarthat. Definiálni kell megállási feltételeket. Ha az építés elegendő hosszú ideje fut, akkor megfigyelhetjük, hogy egyes csúcsok befoka számottevően nagyobb a többinél. Olyan megállási próbáltunk alkalmazni, melyekkel a LON-ban megjelennek kiemelkedő élsúlyok. Ezen feltételek a következők:

– azon pontok száma meghaladja a θ_v értéket, melyek bemenő élek számának θ_e-szorosa kisebb a csúcshoz tartozó élsúlyok összegénél;

– a felderített pontok száma (amelyet ismerünk a nyilvántartáshoz alkalmazott táblázatból) meghaladja az összes lehetséges seed kombinációk számának θ_M-szeresét;

– ha a LON csúcsainak száma az eredeti G csúcsainak számának kétszeresénél több;

– a LON-ban megjelenő maximális élsúly nagyobb, mint a G éleinek számának negyede;

Kitétel ezen feltételek mellett, hogy az első feltételben említett megfelelő csúcsok száma több legyen mint 10.

5.3.2. *H* gráfok

A LON-ban a befok jellemzi egy lokális optimum vonzáskörzetét. Mivel több olyan pont van, melyekre csak a sztochasztikus lépések vezettek, de ritkán, így ezen valószínűtlen, érdektelen pontokat töröljük a LON-ból. Későbbiekben nevezzük ezt a szűrt hálózatot H-nak. A szűrés során az élsúlyokat vizsgáljuk a következő módon. Az 1 súlyú élektől kezdjük. Amennyiben az adott súlyú élek egymás utáni törlésével 2-nél nagyobb méretű komponensek jelennek meg, akkor nem hajtjuk végre ezen

élek törlését és abbahagyjuk a szűrést. Ha a törlésekkel nem vágtunk le nagyobb komponenst, akkor növeljük a súlyértéket, amely szerint törlünk.

5.4. Numerikus kísérletek

Ebben a szakaszban bemutatunk néhány olyan jellegű elemzést, amely a fentiekben leírt módszerek alapján készült lokális optimumok hálózatára vonatkozik. Célunk tehát, hogy a befolyás terjedés maximalizálás probléma keresési terének értékkészletére adjunk jellemzést.

A kísérletek elvégezéséhez mesterségesen generált (szintetikus) véletlen hálózatokat használtunk. Három típust vizsgáltunk:

- Watts-Strogatz (WS) hálózatokat, amelyeket kisvilág gráfoknak is nevezünk [65]
- Barabási-Albert (BA) hálózatokat, amelyek építéséhez a preferenciális kapcsolódás mechanizmust használjuk, ahol a már magas fokszámú csúcsok nagyobb eséllyel kapnak újabb éleket [2];
- Cooper-Frieze (CF) hálózatokat, amelyek a web gráfok növekedési mechanizmusát hivatottak modellezni [12].

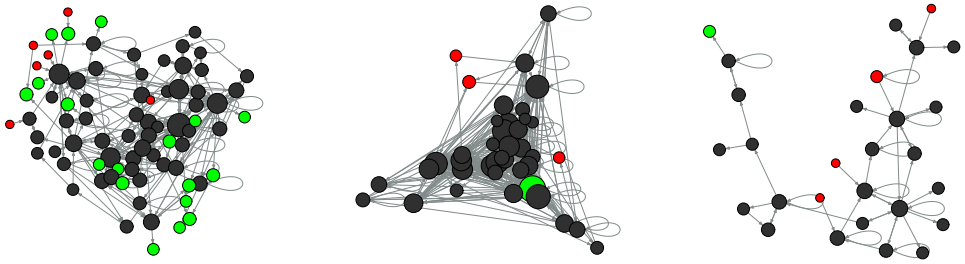
Ezekből a gráfokból több példányt is készítettünk, különböző csúcs- és élszámmal. Kísérletezéseink során azonban arra jutottunk, hogy az $n = 120$ csúcsszámú input gráfokkal kaptunk olyan eredményeket, amelyekben a lokális optimumok hálózatain az élsúlyok nagyobbak voltak 1-nél. Így végül ezekből a gráfokból válogattunk ki olyan eredményeket, amelyek jól reprezentálják az egyes típusokon kapott eredményeket. Az így kiválogatott gráfok élszámai rendre $960, 474$, és 622 voltak a WS, BA és CF hálózatok esetén. Az input gráfok éleihez a súlyokat a $(0, 0.7]$ intervallumon egyenletes eloszlású véletlenszám generátorral rendeltük hozzá.

A futtatásokat a $k = 2$ seed halmaz méretre végeztünk el, a megállási feltételek (lásd 5.3. szakasz) pedig a következők voltak: $\theta_v = \sqrt{|V(G)|}$, $\theta_e = 2$, és $\theta_M = 0.6 \cdot \binom{|V(G)|}{k}$.

5.4.1. *H* gráfok vizsgálata

A 5.2.3. szakaszban bevezetett hegymászó-jellegű eljárás futtatásával kapott lokális optimumok hálózataiból a 5.3. szakaszban ismertetett szűrési eljárással kapjuk meg a H gráfokat. A három különböző input gráf típusból válogatott példákra kapott H gráfokat az 5.1. ábrán láthatjuk[1]. A H gráfok csúcsainak mérete az adott csúcs befokával

[1] szeretnénk hangsúlyozni, hogy itt nem az eredeti G input gráfok vizualizációit láthatjuk



(a) *Watts-Strogatz* **(b)** *Barabási-Albert* **(c)** *Cooper-Frieze*

5.1. ábra. *Lokális optimumok hálózatai, szűrt változatok (H gráfok)*

egyenesen arányos, míg a színek a hamarosan bevezetendő és használt közelség központiság értékeket reprezentálják. Ezekről az ábrákról egyelőre annyi megállapítható, hogy a három különböző input gráfra három különböző szűrt lokális optimumok hálózatát kaptuk, amely azonnal sugallja a befolyás terjedés maximalizálás feladat diverz jellegű viselkedését.

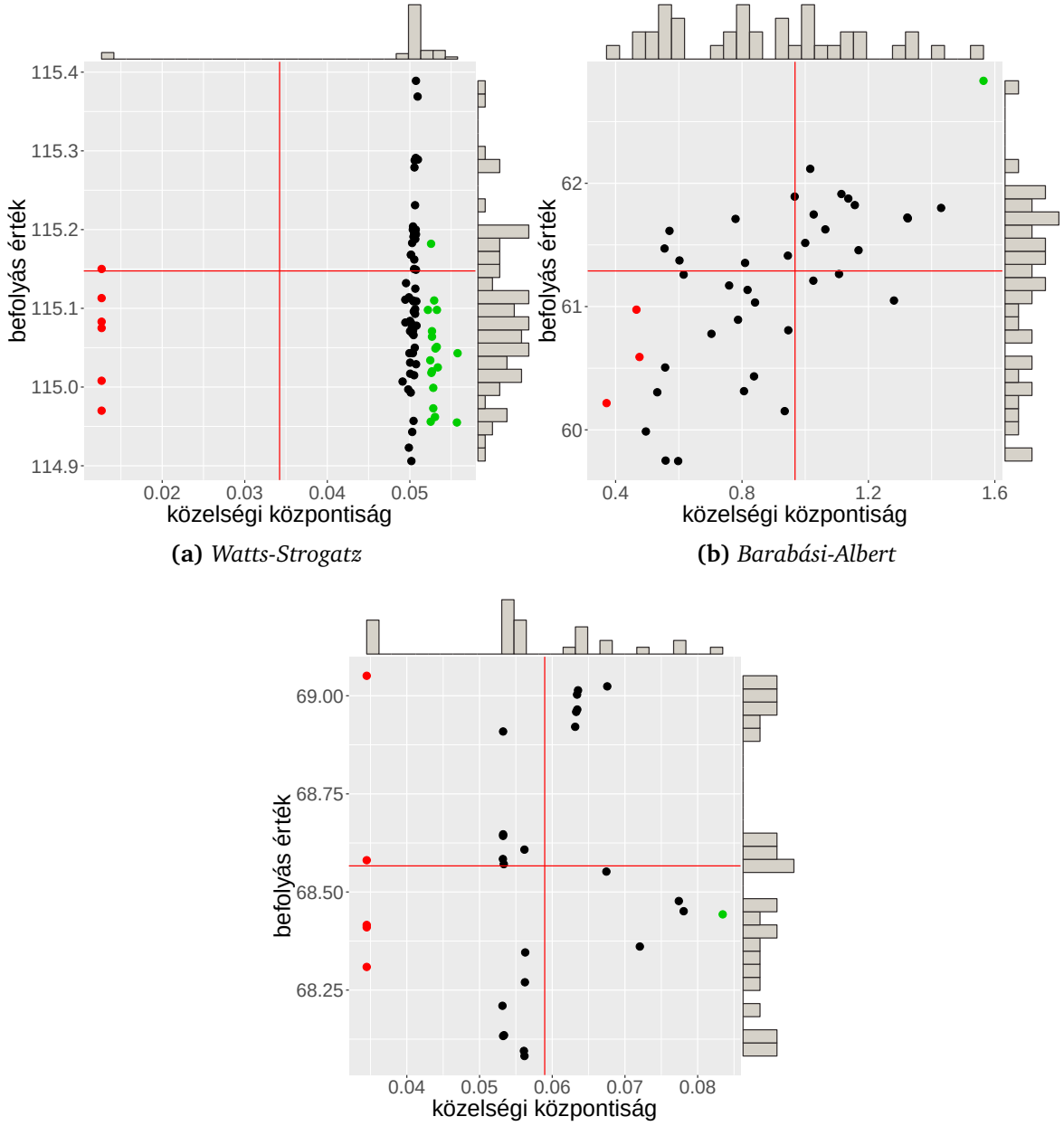
A 5.2. ábrán az egyes H gráfok behatóbb elemzését láthatjuk. A vizsgálati módszer lehetővé teszi, hogy rálátást szerezzünk az egyes lokális optimumok vonzáskörzeteiről. Ehhez a hálózatkutatásban használt közelség központiság (Closeness Centrality) fogalmát használtuk. [2]. Az ábrákon pirossal a CC értékek alsó 10 percentilisébe, zölddel pedig a felső 10 percentilisébe eső csúcsokat színeztük, a többi csúcs pedig feketével van feltüntetve. Ezt a színezést tartottuk meg az 5.1. és 5.4. ábrákon is egyaránt.

A 5.2. ábrán tehát a H gráf csúcsainak közelség központisági értékei és a befolyás értékük közötti összefüggést vizsgálhatjuk. Emlékeztetőül, a H gráf csúcsai az eredeti G input gráf (jelen példákban $k = 2$ elemű) csúcshalmazai. Ez az elemzés lehetővé teszi, hogy egy adott input gráfot megoldhatóság szerint kategorizáljuk könnyűnek vagy nehéznek. A magas befolyásérték jelzi a legjobb talált megoldáshoz való közelséget. Az alacsony CC érték jelenti, hogy az adott pont nehezebben volt elérhető a hegymászó algoritmussal. Amennyiben a pontok többsége magas befolyás értékkel és magas CC értékkel rendelkezik, akkor azt a feladatot könnyűnek nevezhetjük.

Mindhárom input gráfra jellemző, hogy a talált lokális optimumok értékben egymástól kevésben különböznek. A kérdés csak az, hogy vajon ezek mennyire esnek távolra egymástól?

A 5.2a ábra alapján a vizsgált Watts-Strogatz gráf számos nagyon magas értékű

[2]a távolság kiszámításánál az élek súlyainak reciprokát használtuk, követve így azt a konvenciót, hogy magasabb CC érték jelenti azt, hogy az adott csúcs átlagosan közelebb van a többi csúcshoz



(a) *Watts-Strogatz* **(b)** *Barabási-Albert*

(c) *Cooper-Frieze*

5.2. ábra. *Közelségi központiság értékek a H gráfokban*

seed halmazt tartalmaz, amelyek közül a legmagasabb befolyás értéket elérő egyben magas CC értékkel is rendelkezik, ezért annak megtalálása viszonylag könnyű. Ugyanakkor láthatjuk, hogy van egy másik régió is, alacsonyabb fertőzés értéket eredményező és egyben nehezebben elérhető seed halmazokkal.

A 5.2b ábra az előzőtől különbözik abban, hogy a közelség központisági értékek nagyjából egyenletesen oszlanak el. Ez az értékkészletben történő egyenletes szét-

területükre utal. A legmagasabb befolyás értékkel rendelkező seed halmaz egyben a legmagasabb CC értékkel is rendelkezik, így ennek megtalálására a hegymászó algoritmusnak jó esélye van.

A 5.2c ábra a másik két ábrától annyiban különbözik, hogy négy, befolyás értékben minimálisan különböző seed halmaz jelenlétét mutatja. Ezek közül az abszolút legjobb értékű egyben a legnehezebben elérhető is.

5.4.2. Csúcshalmazok egy távolsága: *VSD*

A keresés során alkalmas seed halmazok megtalálása a cél. Ahhoz, hogy az így megtalált halmazok közötti összefüggéseket vizsgáljuk, bevezetünk egy csúcshalmazok távolságának kifejezésére alkalmas mértéket, amelyet *VSD*-vel (*Vertex Set Distance*) jelölünk. Formálisan, legyenek V_1 és V_2 a G gráf csúcshalmazának k-elemű részhalmazai. Legyen

$$VSD(v_1, v_2) = \min_{y \in S_k} \left(\sum_{i=1}^{k} d(V_{1_i}, V_{2_{y_i}}) \right),$$

ahol $d : V(G) \times V(G) \to \mathbb{N}_0$, a gráfon vett távolság, és S_k az első k természetes számból álló permutációk halmaza.

5.1. Tétel. *A* VSD *függvény nemnegatív, szimmetrikus és teljesíti a háromszög egyenlőtlenséget, vagyis matematikai értelemben vett távolságfüggvény.*

Bizonyítás. A *VSD* irányítatlan gráfokon értelmezett legrövidebb utakon alapszik, ami szimmetrikus. Az eredeti G gráf irányított, de minden élnek van párja az ellenkező irányba és a súlyok figyelmen kívül voltak hagyva (gráf távolság mindkét irányba megegyezik). Gráfbeli távolságok összege független a tagok sorrendjétől, ezért *VSD* szimmetrikus, azaz $\textit{VSD}(S_1, S_2) = \textit{VSD}(S_2, S_1)$ teljesül.

Ahogy előzőleg említettük, gráf távolságon alapszik, amelyek értéke mindig nemnegatív. Ezen nemnegatív számok összege is nem negatív. Legyen S_1, S_2 a G gráf csúcsaiból álló, k különböző elemeket tartalmazó halmazok. A $\textit{VSD}(S_1, S_2) = 0$ akkor és csakis akkor, ha $S_1 = S_2$. Ekkor minden S_1 beli elem 0 távolságra van az S_2-beli megfelelőjétől (önmagától), a lehetséges úthosszak összegeinek minimuma pedig 0. Ha S_1 és S_2 egy elemben is térnek el, akkor legalább egy 0-nál hosszabb útnak kell lennie az elemek között, a VSD nagyobb lesz 0-nál, amivel a nemnegativitást beláttuk.

Végül, a háromszög egyenlőtlenséghez a

$$VSD(S_1, S_2) \leq VSD(S_1, S_3) + VSD(S_2, S_3)$$

egyenlőtlenségnek teljesülnie kell $V(G)$ bármely k elemű S_1, S_2, S_3 részhalmazára. Az indirekt bizonyításhoz tegyük fel, hogy $\textit{VSD}(S_1, S_2) > \textit{VSD}(S_1, S_3) + \textit{VSD}(S_2, S_3)$.

Mivel *VSD* szimmetrikus, ezért ez csak akkor teljesülhet, ha

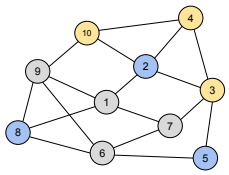
$$VSD(S_1, S_2) > VSD(S_1, S_3) + VSD(S_3, S_2)$$

is teljesül. Tudjuk, hogy

$$VSD(S_1, S_3) = \sum_{i=1}^{k} d(S_{1_i}, S_{3_j}).$$

Hasonlóan megfogalmazható az S_3, S_2 közötti *VSD* érték is. Ekkor vannak utak az S_1 és S_2 elemei között. Ezen utak közül válasszuk azokat, amelyeket a *VSD* is választana. A hosszuk összege legyen D (pont az SVD szerinti távolság). A definícióból adódik, hogy ezen D minimális, meghatározott csúcsok érintésének kikötésével csak növekedhet. Vagyis $D \leq$ *VSD*$(S_1, S_3)+$ *VSD*(S_3, S_2). Ugyanakkor $D =$*VSD*(S_1, S_2), aminek nagyobbnak kell lennie a két előző *VSD* érték összegénél, amely ellentmondás. □

A *VSD* kiszámítását úgy végezzük, hogy egy hozzárendelési feladatként értelmezzük a problémát. Az egyik seed halmaz elemeit szeretnénk hozzárendelni a másik seed halmaz elemeihez. A költségek mátrixa az eredeti G gráf súly nélküli változatából számított távolság részmátrix lesz, amely csak az érintett elemekhez kapcsolódó értékekből áll. Az irányítottságtól eltekinthetünk, mert minden élhez létezik annak ellenkező irányú párja. A megoldás értéke lesz a *VSD*.



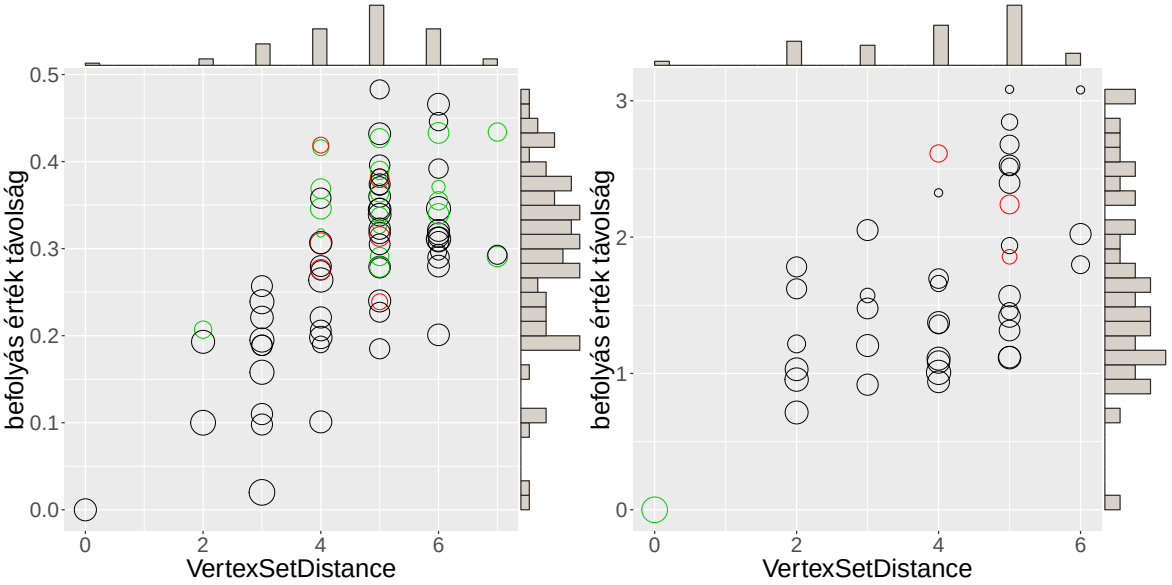
d	3	4	10
2	1	2	$\underline{1}$
5	$\underline{1}$	2	3
8	3	$\underline{3}$	2

5.3. ábra. VSD *kiszámítása hozzárendelési feladatként*

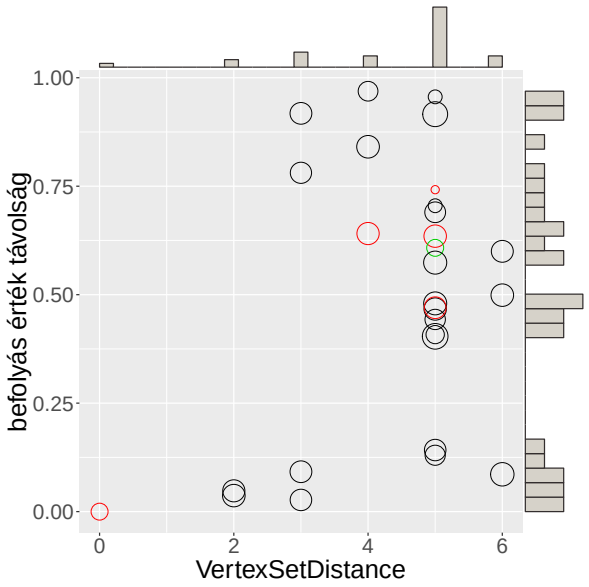
Példa. Egy könnyen áttekinthető példát a 5.3. ábrán láthatunk, ahol G egy 10 csúcsból álló gráf, a két csúcshalmaz pedig $S_1 = \{2, 5, 8\}$ (az ábrán kék színnel jelölve) és $S_2 = \{3, 4, 10\}$ (az ábrán sárga színnel jelölve). A kapcsolódó hozzárendelési feladatot a jobb oldali táblázat írja le, amiből a *VSD*$(S_1, S_2) = 5$ megoldás adódik.

Az így bevezetett *VSD* távolság mértéket használhatjuk annak a kielemzésére, hogy a megtalált lokális optimumok közül a legmagasabb értékűhöz viszonyítva a többiek (*VSD* értelemben) milyen távol helyezkednek el az eredeti G gráfban. Ezeket az értékeket a legjobb megoldás befolyás értékétől vett távolsággal vetjük össze.

A 5.4. ábrán a három input gráfra kapott eredményeket láthatjuk. A körök mérete az adott seed halmaz *vonzáskörzetével* arányos: a hegymászó, mint lokális optimalizáló futtatása során hányszor jutottunk egy lokális optimumba. Az origóban tehát az



(a) *Watts-Strogatz* **(b)** *Barabási-Albert*



(c) *Cooper-Frieze*

5.4. ábra. VSD *értékek a* G *input gráfokon*

adott G gráfban talált legjobb megoldás helyezkedik el, a távolságokat ehhez a seed halmazhoz viszonyítjuk. A 5.4b ábra szerint a BA hálózat (ezen példányának) értelmezési tartománya ú.n. *big-valley* szerkezetű [5]. Ehhez hasonlít a 5.4a ábrán látható WS hálózat is, bár abban találtunk egy, a legjobb megoldáshoz nagyon közeli befolyás értékű seed halmazt is. Végül a 5.4c ábrán újra igazolást nyer a CF hálózatok a másik két típustól vett különbözősége, hiszen itt összesen hét darab magas értékű lo-

kális optimumot találtunk, amelyek az egyébként nehezen megtalálható (erre a piros színezés utal) legjobb megoldástól, VSD értelemben, egyre távolabb találhatóak.

5.5. Konklúzió

A szakirodalomban sok szempontból vizsgált, erősen alkalmazás orientált problémát, a gráfokon értelmezett befolyás terjedés maximalizálás megoldásainak tulajdonságait vizsgáltuk. Megközelítésünk újdonságát az adta, hogy a felhasznált hegymászó-jellegű algoritmussal a keresési tér olyan leképezését vizsgáltuk, amely lehetővé teszi a feladat strukturális tulajdonságainak mélyebb megértését. Ehhez felhasználtuk a lokális optimumok hálózatának (LON) fogalmát. Ennek segítségével három, szerkezetileg különböző véletlen gráfon, mint inputon kimutattuk, hogy a probléma különböző értékkészlet struktúrákhoz vezet. Az így kapott eredmények lehetőséget adnak célzott keresési eljárások definiálására, ennek részletes kidolgozását jövőbeli kutatási tervként jelöljük ki.

Összefoglalás

Az optimalizálás és a gráfelmélet illetve a hálózattudomány sok tudományterületen felhasználható. A matematika különböző területein, a biológiában, mérnöki tudományokban és a szociológiában is megjelennek olyan problémák, melyekhez hatékony eszközöket adnak.

A disszertáció gráf problémákkal vagy gráfokat alkalmazó megoldásokkal foglalkozik. A két fő témakörhöz kettő-kettő fejezet tartozik. Az első témakör a gráfok rekonstukciójáról szól. A második témakör a gráfok használata az optimalizálásban taglalja. A hozzá tartozó fejezetekben a Lokális Optimumok Hálozata eszközt használtuk.

Gráf rekonstrukció

A 2. fejezet a szakirodalomban régóta ismert gráf rekonstrukció azon változatával foglalkozik, ahol az ismert adataink a csúcsok Közelség centralitás értékei és a rekonstruálandó gráf fa.

A probléma azért érdekes, mert felhasználható korlátozott adatokból való hálózatok helyreállításához, adat tömörítéshez vagy generatív modellek készítéséhez, melyek teszteléseket segíthetik.

Ezen munkában a jelenleg elterjedt populáció-alapú, sztochasztikus algoritmusoktól eltérő eszközt szerettünk volna alkotni. A Közelség központiságból való rekonstrukciót tudomásunk szerint nem vizsgálták a munkánk befejeztéig. Célunk volt ezen specializált feladat megértése és olyan összefüggések megtalálása, melyekkel segíthetjünk az általános esetek megoldását.

Több algoritmust alkottunk és teszteltünk. Észrevételeink egyike volt a gráfban szomszédos csúcsok értékei közötti összefüggés. Empirikus erdeményekből az tapasztaltuk, hogy a fák fokszámsorozat és a rekonstruálási feladat nehézzsége (ezen algoritmusokkal) között kapcsolat van.

A Maximum Presolve algoritmusunk gyorsan elkészülő és közel jó megoldásival más algoritmusok kezdő értékeinek kiválasztását segíthetjük. A populáció-alapú algoritmusokban való felhasználása jövőbeli tervünk.

A Recursive algoritmusunk pontos megoldások keresésre lett fejlesztve. Az 1000

méretű gráfok tesztelésénél azt tapasztaltuk, hogy bizonyos tulajdonsággal rendelkező gráfok esetén nem hatékony. További összefüggéseket szeretnénk keresni, melyekkel a futási idő csökkenne ilyen gráfokon.

A szerző hozzájárulása

A disszertáció szerzője alkotta meg és implementálta a fejezetben ismertett algoritmusokat. A tesztelések megtervezése és kiértékelése, a fejezetben szereplő eredmények cikk formájú leírása közös munka a témavezetővel.

Gráf realizáció

A 3. fejezet a rekonstrukciónál egyszerűbb feladatot tárgyal, a realizációt. A feladat csak az, hogy eldöntsük az adott adatokból építhető-e gráf. Az előzőhöz fejezethez hasonlóan közelségi centralitás értékekből dolgozunk.

A munka során fa és nem-fa gráfokra sikerült 4-4 feltételt alkotni. Ezen feltételek csak szükségesek, így a feltételeket teljesítő adatsorból nem garantált a pontos egyezésű gráf alkotása. A feltételeket bizonyítottuk is.

A 2. fejezetben szereplő probléma gyors input tesztelésére használhatóak ezek a feltételek. Így elkerülhető, hogy költségesebb algoritmusokat kelljen használni nem megfelelő adatokon.

Az első feltétel az inputban szereplő értékek összegének paritását szabja meg. A második feltétel az input értékekre ad alsó és felső korlátot. A harmadik az inputban szereplő értékek lehetséges tartományát adja meg. A negyedik az input értékek összegére ad alsó és felső korlátot.

A szerző hozzájárulása

A disszertáció szerzője fogalmazta meg a fejezetben ismertetett feltételeket és dolgozta ki azok bizonyításait.

Memetic Differential Evolution hálózattudományi eszközökkel

A 4. fejezet a Memetic Differential Evolution populáció-alapú algoritmus vizsgálatáról és fejlesztéséről szól. Néhány közismert tesztfüggvény globális optimumát kerestük.

Az eredeti algoritmusban szereplő szabályt megváltoztattuk. A megtalált lokális optimumokból épített hálózatbeli centrealitási értékeket használtuk az egyedek kiválasztására. A különböző szabáylokkal különböző hálózatok készültek.

Az eredeti algoritmust kezdtük viszgálni és az általa gyártott hálózatokat. A további teszteléseink során nagyobb dimenzió méretet, eltolást és forgatást használtunk, hogy a tesztelendő függvények globális optimumát nehezebb legyen megtalálni. A globális optimum értékeket ismertük.

A közismert nyolc MDE változat futtatása közben épített hálózatokat felhasználva egyszerű újraindítási szabályt fogalmaztunk meg a korai konvergencia okozta hiba ellen, amivel bonyolúltabb problémáknál javulást értünk el.

Az első vizsgálatainkhoz hasonlóan a futás közben épített hálózatban lévő centralitás értékeket használtuk új szabályokhoz. Ezeket a szabályokat hasonlítottuk össze az eredeti klasszikussal ugyanazon bonyolúltabb tesztfüggvényeken, melyeket a fejezett korábbi részében használtunk.

A felhasznált centralitási mértékek a közelség, a közöttiség, a fokszám, a harmonikus centralitás és a PageRank volt. Az adott paraméterekkel a klasszikus MDE bizonyúlt az egyik leghatékonyabbnak.

A szerző hozzájárulása

A fejezetben bemutatott algoritmusok implementálását, az utolsó alfejezetben szereplő tesztekhez szükséges kiegészítések kivételével, és a tesztek elvégzését a szerző hajtotta végre.

A tesztek tervezése, az eredmények értelmezése és a fejezetben bemutatott Above-Below szabály a témavezető és a szerző közös munkája.

Befolyás terjedés maximalizálás probléma terének vizsgálata

Az 5. fejezet egy gráf-alapú optimalizálási probléma vizsgálatáról szól, melynek neve Befolyás terjedés maximalizálás. A probléma lokális optimumait derítettük fel és alkottunk belőlük hálózatot, hasonlóan az előző fejezetbelihez.

Célunk a probléma megértése volt és olyan összefüggések találása, melyekkel a problémához tudunk előfeldolgozóknak vagy egyéb módszereknek segítő észrevételeket megfogalmazni.

A közismert Monte Carlo-módszert alkalmazó terjedési modellek közül a Független Kaszkádokat (Independent Cascades) vizsgáltuk. Megfogalmaztunk lokális optimumokat ebben a diszkrét problémában, melyekből építettünk egy az előző fejezetben bemutatott Lokális Optimumok Hálózatától eltérőt. Ezen hálózat élei súlyozottak

és írányítottak. A definíciójához igazodva alkottunk egy hegymászó-jellegű algoritmust.

Az algoritmusunk nem a probléma megoldását kereste. Egy-egy optimális megoldás megtalálásához a szakirodalomban ismert a Befolyás terjedés Maximalizálás problémához tartozó Mohó algoritmust használtuk. Az algoritmusunk terjedési modellt ismételte az előző iteráció eredményétől függő kezdőértékkel az általunk megfogalmazott feltételek teljesüléséig.

A munka során az általunk ebben a részben megfogalmazott Lokális Optimumok Hálózatának diszkrét elemeket tartalmazó csúcsaira definiáltunk távolság függvényt, melyet Vertex Set Distance-nek neveztünk el (röviden VSD). Viszgálatokat végeztünk különböző generatív modellekkel készített gráf típusokon, melyeket használtuk fel inputként. A hegymászó-jellegű algoritmusunk által adott hálózatokat egyszerűsítettük, bizonyos felesleges elemeket töröltünk. Végül a diszkrét probléma terét vizualizáltuk további kisérletek segítéséért.

A szerző hozzájárulása

A fejezetben ismertetett hegymászó-jellegű algoritmus megalkotása, a szakirodalomban ismert terjedési modell kivételével, a témavezető és a szerző közös munkája, a speciális Lokális Optimumok Hálózatának megfogalmazásával, a tesztek tervezésével és az eredmények értelmezésével együtt.

A szerző végezte el az implementálásokat, teszteléseket, a VSD megfogalmazását és bizonyítását arról, hogy valódi távolság fogalom.

Summary

Optimization, graph theory and network science can be used in many scientific fields. In various fields of mathematics, biology, engineering and sociology, problems appear for which they offer effective tools.

The dissertation deals with graph problems and solutions using graphs. There are two chapters each for the two main topics. The first topic is about the reconstruction of graphs. The second topic covers the use of graphs in optimization. In the corresponding chapters, we used the Local Optima Network tool.

Graph reconstruction

Chapter 2 deals with the version of graph reconstruction that has long been known in the literature, where the known data are the Closeness Centrality values of the vertices and the graph to be reconstructed is a tree.

The problem is interesting because it can be used to reconstruct networks from limited data, to compress data, or to create generative models that can aid testing.

In this work, we wanted to create a tool different from the currently widespread population-based, stochastic algorithms. To our knowledge, the reconstruction from the Closeness Centrality was not examined until the completion of our work. Our goal was to understand this specialized task and to find connections with which we could help solve general cases.

We created and tested several algorithms. One of our observations was the correlation between the values of adjacent vertices in the graph. From empirical results, we found that there is a relationship between the degree sequence of the trees and the difficulty of the reconstruction task (with these algorithms).

With the fast and good solutions of our Maximum Presolve algorithm, we can help the selection of the starting values of other algorithms. Our future plan is to use it in population-based algorithms.

Our Recursive algorithm was developed to find exact solutions. When testing graphs of size 1000, we found that it is not effective for graphs with certain properties. We would like to look for additional relationships that would reduce the running time on such graphs.

The contributions of the author

The author of the dissertation created and implemented the algorithms described in the chapter. The planning and evaluation of the tests and the description of the results in the chapter in the form of an article are joint work with the supervisor.

Graph realization

Chapter 3 discusses a simpler task than reconstruction, realization. The only task is to decide whether a graph can be built from the given data. Similar to the previous chapter, we work from Closeness Centrality values.

During the work, it was possible to create 4 conditions for tree and non-tree graphs. These conditions are only necessary, so the creation of an exact matching graph is not guaranteed from the data sequence that fulfills the conditions. We have also proved the conditions.

These conditions can be used for rapid input testing of the problem in Chapter 2. This avoids having to use more expensive algorithms on inappropriate data.

The first condition determines the parity of the sum of the values included in the input. The second condition gives lower and upper bounds to the input values. The third specifies the possible range of values in the input. The fourth gives a lower and upper bounds for the sum of the input values.

The contributions of the author

The author of the dissertation formulated the conditions described in the chapter and elaborated their proofs.

Memetic Differential Evolution with Network Science tools

Chapter 4 is about the investigation and development of the population-based algorithm Memetic Differential Evolution (MDE). We searched for the global optimum of some well-known test functions.

We have changed the rule in the original algorithm. The network centrality values constructed from the found local optima were used to select the individuals. Different networks were created with different rules.

We began to examine the original algorithm and the networks it produced. During our further testing, we used a larger dimension size, shift and rotation to make it

more difficult to find the global optimum of the functions to be tested. We know the global optimum values.

Using the networks built while running the eight well-known MDE versions, we formulated a simple restart rule against the error caused by early convergence, with which we achieved an improvement in more complicated problems.

Similar to our first tests, we used the centrality values in the network built at runtime for new rules. We compared these rules with the original classic on the same more complicated test functions that we used in the previous part of the chapter.

The centrality measures used were closeness, betweenness, degree, harmonic centrality and PageRank. With the given parameters, the classic MDE proved to be one of the most effective.

The contributions of the author

The implementation of the algorithms presented in this chapter, with the exception of the additions required for the tests in the last subsection, and the execution of the tests were carried out by the author.

The planning of the tests, the interpretation of the results and the Above-Below rule presented in the chapter are the joint work of the supervisor and the author.

Investigation of the space of Influence Maximization Problem

Chapter 5 deals with the investigation of a graph-based optimization problem called Influence Maximization. We discovered the local optima of the problem and created a network from them, similar to the one in the previous chapter.

Our goal was to understand the problem and to find connections with which we can formulate helpful ideas for preprocessors or other methods.

Among the propagation models using the well-known Monte Carlo method, we examined the Independent Cascades. We formulated local optima in this discrete problem, from which we built a Local Optima Network different from the one in the previous chapter. The edges of this network are weighted and directed. Adapting to its definition, we created a hill-climbing algorithm.

Our algorithm was not looking for a solution to the problem. In order to find an optimal solution, we used the Greedy algorithm known in the literature for the Influence Maximization problem. Our algorithm repeated the propagation model with a starting value depending on the result of the previous iteration until the conditions we formulated were met.

During the work, we defined a distance function for the vertices containing discrete elements of the Local Optima Network formulated in this section, which we named Vertex Set Distance (VSD for short). We performed studies on graph types created with different generative models, which were used as input. We simplified the networks given by our hill-climbing algorithm and deleted certain uninteresting elements. Finally, we visualized the space of the discrete problem to help further experiments.

The contributions of the author

The creation of the hill-climbing algorithm described in the chapter, with the exception of the propagation model known in the literature, is the joint work of the supervisor and the author, together with the formulation of the special Local Optima Network, the design of the tests and the interpretation of the results.

The author carried out the implementations, testing, formulation of the VSD and proof that it is a distance function (or metric).

Köszönetnyilvánítás

Elsősorban megköszönöm témavezetőmnek, Dr. Vinkó Tamásnak, a több éves közös munkát, tanulási lehetőséget és segítségeket, melyek nélkül nem készült volna el ez a disszertáció.

Szeretném megköszönni kollégáimnak a segítő szakmai tanácsokat és ötleteket a munkáimhoz.

Hálás vagyok családomnak és barátaimnak a rengeteg támogatásért.

Irodalomjegyzék

[1] C. C. Aggarwal, S. Lin, and P. S. Yu. On influential node discovery in dynamic social networks. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 636–647. SIAM, 2012.

[2] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[3] D. Barmpoutis and R. M. Murray. Networks with the smallest average distance and the largest average clustering. *arXiv preprint arXiv:1007.4031*, 2010.

[4] A. Bavelas. Communication patterns in task-oriented groups. *The journal of the acoustical society of America*, 22(6):725–730, 1950.

[5] K. D. Boese, A. B. Kahng, and S. Muddu. On the big valley and adaptive multi-start for discrete global optimizations. Technical Report TR-930, 15, University of California, 1993.

[6] F. Cabassi and M. Locatelli. Computational investigation of simple memetic approaches for continuous global optimization. *Computers & Operations Research*, 72:50 – 70, 2016.

[7] E. Candes and B. Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[8] P.-A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning*, pages 80–95. Springer, 2003.

[9] P. Ćirković, P. Đorđević, M. Milićević, and T. Davidović. Metaheuristic approach to spectral reconstruction of graphs. In *International Conference on Mathematical Optimization Theory and Operations Research*, pages 79–93. Springer, 2022.

[10] F. Comellas and J. Diaz-Lopez. Spectral reconstruction of complex networks. *Physica A: Statistical Mechanics and its Applications*, 387(25):6436–6442, 2008.

[11] F. Comellas and J. Paz-Sánchez. Reconstruction of networks from their betweenness centrality. In *Workshops on Applications of Evolutionary Computation*, pages 31–37. Springer, 2008.

[12] C. Cooper and A. Frieze. A general model of web graphs. *Random Structures & Algorithms*, 22(3):311–335, 2003.

[13] G. Csardi, T. Nepusz, et al. The igraph software package for complex network research. *InterJournal, complex systems*, 1695(5):1–9, 2006. `https://igraph.org`.

[14] F. Daolio, M. Tomassini, S. Vérel, and G. Ochoa. Communities of minima in local optima networks of combinatorial spaces. *Physica A: Statistical Mechanics and its Applications*, 390(9):1684–1694, 2011.

[15] K. Das, S. Samanta, and M. Pal. Study on centrality measures in social networks: a survey. *Social Network Analysis and Mining*, 8:1–11, 2018.

[16] P. Domingos and M. Richardson. Mining the network value of customers. In *In Proceedings of 7th International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[17] J. K. Doyle and J. E. Graver. Mean distance in a directed graph. *Environment and Planning B: Planning and Design*, 5(1):19–29, 1978.

[18] D. Erdős, R. Gemulla, and E. Terzi. Reconstructing graphs from neighborhood data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):1–22, 2014.

[19] P. L. Erdős, Z. Király, and I. Miklós. On the swap-distances of different realizations of a graphical degree sequence. *Combinatorics, Probability and Computing*, 22(3):366–383, 2013.

[20] P. Erdős and T. Gallai. Gráfok eloírt fokú pontokkal (graphs with points of prescribed degrees, in hungarian). *Mat. Lapok*, 11:264–274, 1961.

[21] R. Fourer and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2002.

[22] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[23] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Celf++ optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48, 2011.

[24] A. Hagberg, P. Swart, and D. S. Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2008.

[25] L. Hajdu, M. Krész, and A. Bóta. Community based influence maximization in the independent cascade model. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 237–243. IEEE, 2018.

[26] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3) :496–506, 1962.

[27] S. L. Hakimi and S. S. Yau. Distance matrix of a graph and its realizability. *Quarterly of applied mathematics*, 22(4) :305–317, 1965.

[28] W.E. Hart, C.D. Laird, J.P. Watson, D.L. Woodruff, G.A. Hackebeil, B.L. Nicholson, and J.D. Siirola. *Pyomo-optimization modeling in python*. Springer, 2017. SOIA, volume 67.

[29] V. Havel. A remark on the existence of finite graphs. *Casopis Pest. Mat.*, 80 :477–480, 1955.

[30] J. J. Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of mathematical biology*, 51(5) :597–603, 1989.

[31] V. Homolya and T. Vinkó. Memetic differential evolution using network centrality measures. In *Proceedings of the 14th International Global Optimization Workshop*, pages 250–254. AIP Publishing, 2019.

[32] V. Homolya and T. Vinkó. Befolyás terjedés optimumainak hálózatáról. *Alkalmazott Matematikai Lapok*, 37 :167–179, 2020.

[33] V. Homolya and T. Vinkó. Leveraging local optima network properties for memetic differential evolution. In *Optimization of Complex Systems : Theory, Models, Algorithms and Applications*, pages 109–118. Springer-Verlag, 2020.

[34] V. Homolya and T. Vinkó. Closeness centrality reconstruction of tree graphs. *Central European Journal of Operations Research*, (közlésre elfogadva) 2023.

[35] M. Ipsen and A. S. Mikhailov. Evolutionary reconstruction of networks. *Physical Review E*, 66(4) :046109, 2002.

[36] P. J. Kelly. A congruence theorem for trees. 1957.

[37] D. Kempe, J Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[38] H. Kim, P. L. Toroczkai, Z.and Erdős, I. Miklós, and L. A. Székely. Degree-based graph construction. *Journal of Physics A: Mathematical and Theoretical*, 42(39):392001, 2009.

[39] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[40] J.-R. Lee and C.-W. Chung. A fast approximation for influence maximization in large social networks. In *Proceedings of the 23rd international conference on World Wide Web*, pages 1157–1162, 2014.

[41] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2007.

[42] M. Locatelli, M. Maischberger, and F. Schoen. Differential evolution methods based on local searches. *Computers & Operations Research*, 43:169–180, 2014.

[43] M. Lozano and F. J. Rodriguez. Network reconstruction from betweenness centrality by artificial bee colony. *Swarm and Evolutionary Computation*, 62:100851, 2021.

[44] M. Marchiori and V. Latora. Harmony in the small-world. *Physica A: Statistical Mechanics and its Applications*, 285(3-4):539–546, 2000.

[45] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[46] N. Metropolis. The beginning. *Los Alamos Science*, 15:125–130, 1987.

[47] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 485–492, 2006.

[48] J. J. Moré and T. S. Munson. Computing mountain passes and transition states. *Mathematical Programming B*, 100:151–182, 2004.

[49] P. Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826(1989):37, 1989.

[50] B.A. Murtagh and M.A. Saunders. Minos 5.5.1 user's guide. Technical report, 2003.

[51] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[52] M.E.J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[53] L. Page, S. Brin, R. Motwani, and T. Winograd. Pagerank: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper, 1997.

[54] P. K. Pandey and B. Adhikari. A parametric model approach for structural reconstruction of scale-free networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2072–2085, 2017.

[55] A.P. Piotrowski. Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. *Information Sciences*, 241:164–194, 2013.

[56] K. Rahimkhani, A. Aleahmad, M. Rahgozar, and A. Moeini. A fast algorithm for finding most influential people based on the linear threshold model. *Expert Systems with Applications*, 42(3):1353–1361, 2015.

[57] A. Saxena and S. Iyengar. Centrality measures in complex networks: A survey. *arXiv preprint arXiv:2011.07190*, 2020.

[58] L. Skanderova and T. Fabian. Differential evolution dynamics analysis by complex networks. *Soft Computing*, 21(7):1817–1831, 2017.

[59] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[60] A. Tripathi and S. Vijay. A note on a theorem of Erdős & Gallai. *Discrete Mathematics*, 265(1-3):417–420, 2003.

[61] S. M. Ulam. *A collection of mathematical problems*. Interscience Tracts in Pure and Applied Mathematics, no. 8. New York, Interscience, 1960.

[62] L. G. Varga, L. G. Nyúl, A. Nagy, and P. Balázs. Local and global uncertainty in binary tomographic reconstruction. *Computer Vision and Image Understanding*, 129:52–62, 2014.

[63] T. Vinkó and K. Gelle. Basin hopping networks of continuous global optimization problems. *Central European Journal of Operations Research*, 25:985–1006, 2017.

[64] N. Vuokko and E. Terzi. Reconstructing randomized social networks. In *Proceedings of the 2010 siam international conference on data mining*, pages 49–59. SIAM, 2010.

[65] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684) :440–442, 1998.

[66] J. W. Wright. The change-making problem. *Journal of the ACM (JACM)*, 22(1) :125–128, 1975.

[67] J. H. Yin and J. S. Li. Two sufficient conditions for a graphic sequence to have a realization with prescribed clique size. *Discrete mathematics*, 301(2-3) :218–227, 2005.