**3. Eljárások**

A hálózatok közösségszerkezetének felderítése a hálózattudomány és a gráfelmélet egyik alapvető témájává vált, és a tudomány széles területén alkalmazzák. A probléma iránti nagy érdéklődés miatt számos különféle algoritmus született a hálózati közösségek azonosítására. Ezek közül a szakdolgozatom egy lineáris programozási megközelítést és egy az utazó ügynök problémára alapuló módszert mutat be, és hasonlít össze.

**3.2. TSP alapú (ha van 2 heurisztika a tsp megoldására, akkor azok rövid ismertetésével együtt)**

Az *utazó ügynök probléma (Travelling Salesman Problem, TSP)* az egyik legjobban kutatott probléma az optimalizálás területén, [1] – ben javasoltak a TSP-re alapuló közösségkereső algoritmust. Az utazó ügynök problémacélja, hogy adott városból indulva, adott N város halmazán találjuk egy minimális költségű utat úgy, hogy minden várost pontosan egyszer látogatunk meg, és visszatérünk a kezdő városba. Bármely két város közötti távolság ismert. Utazó ügynök problémaként modellezték a közösségkeresési problémát, egy csúcsot egy városként tekintve. Az ötlet abból ered, hogy mivel egy minimális költségű utat keresünk, az egymáshoz közeli városok általában klasztereződnek az útvonalon. A megoldásként kapott út néhány kisebb útra való szétdarabolásával kaphatjuk meg a hálózat közösségszerkezetét.

**3.2.1. PRF mátrix: PageRank Feature**

Az utazó ügynök problémában a városok közötti távolság előre meghatározott, ezért a közösségkeresési probléma modellezéséhez definiálni kell az egyes csúcspárok közötti távolságot. A távolság azt jelzi, hogy mekkora a valószínűsége, hogy a két csúcs egy közösséghez tartozik. Minél kisebb a távolság, annál nagyobb a valószínűsége, hogy azonos közösségbe kerültek.

A Google keresőmotorjának PageRank funkciója [2] alapján tervezték meg a PageRank Feature-t, amellyel ki tudjuk számítani a megfelelő távolságot. A PageRank egy weboldal rangsoroló algoritmus, amely a Web hivatkozáshálózatának felhasználásával rekurzívan számítja ki az oldalak centralitását. Egy weboldalnak akkor magas a PageRank értéke, ha sok más oldal mutat rá, vagy a rá mutató weboldalak magas PageRank-kel lettek minősítve.

Tegyük fel, hogy van egy *N* oldalból álló weboldal-hálózat, ahol a felhasználó az oldalakon lévő hiperhivatkozásokkal lép az egyik oldalról a másikra. Annak a valószínűségét, hogy t lépés után a pi oldalra érkezünk a következőképpen definiálták [1]-ben:

, ahol annak a valószínűsége, hogy *pj*-ből *pi*-be lépünk, és a *d*∈[0, 1] paraméter egy kiugró faktor. Erre azért van szükség, hogy a felhasználó ne ragadjon hivatkozás nélküli oldalon, és mert a felhasználó nem mindig hiperhivatkozással lép tovább, hanem egy random oldalon folytatja a keresést. Egy másik oldalt a felhasználó *d* valószínűséggel látogat meg hiperhivatkozás használatával, és 1-*d* valószínűséggel hiperhivatkozás nélkül. A PageRank kezdőértéke egy *N* oldalt tartalmazó hálózaton:

Számos iteráció után egy konkrét értékhez konvergál, ez lesz az oldalak PageRank értéke, amely valójában azt mutatja meg, hogy egy felhasználó mekkora valószínűséggel látogat meg egy weboldalt. A PageRank értékek a weboldalak valószínűségi eloszlását alkotják, az összegük egy lesz.

A TSP algoritmusok futtatásához, és ezzel hálózat szerkezetének meghatározásához azonban az egyes csúcspárok közötti távolságra van szükségünk, amelyet a *PRF mátrix* elemeiből tudunk kiszámítani. A PRF mátrix *i*-edik oszlopvektora a *vi* csúcshoz a *t* lépés utáni végpontok valószínűségi eloszlása, az alábbi képlettel definiálták:

, ahol , *j*=1, 2, …, *N* annak a valószínűsége, hogy a *vi* csúcsból indulunk, és a *vj* csúcsban állunk meg *t* lépés után.

A közösség definíciója szerint a hálózatokban a közösségen belüli csúcsok között sűrűbben helyezkednek el élek, mint a közösségek között. Ha a hálózat *Ci* közösségének *vi*∈*V* csúcsából indul az ügynök, akkor nagyobb a valószínűsége, hogy *t* lépés után egy ugyanabban a közösségben lévő csúcsban áll meg, mint egy másik közösségben. Ahhoz, hogy megkapjuk a csúcspárok közötti távolságot, a PRF(*vi*)-nek annak a valószínűségét kell megmutatnia, hogy melyik csúcsban állunk meg, ha a kezdő csúcs egy meghatározott *vi* pont. A PageRank-kel ellentétben, PRF nem csak a végpontot veszi figyelembe, számít, hogy melyik csúcsból indítjuk az utat. Tehát, ha *vi* a megadott kezdő csúcs, akkor *PRF*(*vi*) kezdőértéke (*PRF0*(*vi)* ) egy vektor, ahol az i-edik elem 1, minden más elem 0.

Azt mondtuk, hogy a közeli csúcsok klasztereződnek, ezért a csúcsok távolságából szeretnénk megállapítani a közösségszerkezeteket. Mivel a *PRF* vektorok elemeit használjuk a távolságok kiszámítására, az azonos közösségben lévő csúcsok PRF vektorainak hasonlónak kellene lennie, és különböznie kellene különböző közösségben lévő csúcsokétól. A véletlen séta során *vi* csúcsból indulva hasonló valószínűséggel érkezünk vk (*k*=1, 2, …, *N)* csúcsba, mint ha a *vj* csúcsból indulnánk, hiszen azonos közösséghez tartoznak.

Ahogy *PR,* úgy *PRF* is egy konkrét értékhez konvergál, ezért, ha *t* elég nagy, akkor *vi* és *vj* csúcsok *PRF*t(*vi*) és *PRF*t(*vj*) értéke majdnem megegyezik, nem lenne alkalmas a közösségek megadására. Ha 0 lépés után állnánk meg, akkor az értékek teljesen különböznének minden csúcs külön közösségbe kerülne, végtelen lépés esetén az összes ugyanannyi lenne, egy közösségben helyezkedne el az összes csúcs. [1]-ben a *t* paraméter értékének a 6-ot választották, ezzel az összes csúcsot lefedve a hálózatban, de még a *PRF* vektorok nem közelítenek túlságosan egymáshoz.

**3.2.2. PRD mátrix: PageRank Distance**

A PRF mátrix kiszámítása után minden csúcs leképezhető egy *N* dimenziós Hilbert térbe, ahol a pontok koordinátáit a csúcs PRF vektora adja meg. Bármely két csúcs távolsága az euklideszi távolsággal kiszámolható, amelynek a definíciója az alábbi:

Az összes csúcspár kiszámításának az időigénye *O*(*N*3). Felfedezték, hogy a hálózatok RWF mátrixának elemei között sok hasonló és kicsi pont van. A számítási költség csökkentésére a PRD mátrix kiszámítása során csak a kiugró értéket vették figyelembe, a triviális pontokat 0-val helyettesítették, így csak a kiugró értékekhez tartozó csúcsok távolságait kell kiszámolni. A tetőpontok halmazát a következőképpen definiálták:

Vagyis azon csúcsok halmaza, ahol a PRF érték a legnagyobb az adott csúcs (*vi*) PRF vektorában. ha i-ből indulunk az a legvalószínűbb, hogy j-ben állunk meg

Egy *N* ponttal rendelkező hálózatban a PRF vektorok száma *N*, tehát legfeljebb *N* eleme lesz a *PP* halmaznak. Előfordulhat, hogy több PRF vektornak is ugyanabban a pontban van a legnagyobb értéke, főleg azoknál a csúcsoknál, amelyek ugyanahhoz a közösséghez tartoznak, ezért általában a PP elemeinek száma jóval kisebb lesz N-nél. A PP pontok halmazába általában a komplex hálózat azon néhány csúcsa kerül, amelyeknek nagy a fokszáma, és nagy valószínűséggel ezekben a csúcsokban fogunk megállni a véletlen séta végén.

Ezek alapján a PRD-t a következőképpen definiálták:

A PP halmazzal lecsökkentették a PRF mátrix dimenzióját a halmaz elemeinek számára. Ez a szám különböző hálózatoknál eltér, azonban gyakran a közösségek számának nagyságrendjében van.

**3.2.3. TSP solver**

A közösségkeresési probléma célja, hogy találjunk modulokat a hálózatokban, ahol a modulon belüli csúcsok sűrűbben vannak összekötve mint a különböző modulokhoz tartozók.

A közösségkeresési probléma megoldásához először használnunk kell egy TSP megoldót, amiből megkapunk egy minimális költségű körutat, amely az összes pontot bejárja. Ehhez bármely TSP algoritmust használhatjuk. Annak ellenére, hogy az utazó ügynök probléma NP-nehéz, számos heurisztika és egzakt módszer született a megoldására.

<https://github.com/fillipe-gsm/python-tsp>

**heurisztikák**

**3.2.4. Automatikus küszöbérték alapú vágás**

Minél kisebb a PRD, vagyis minél kisebb a távolság két csúcs között, annál nagyobb a valószínűsége, hogy ugyanabban a közösségben vannak.

Miután megvan az optimális út, már csak az kérdés, hogyan alakítjuk át az utat egy közösség szerkezetté. Meg kell határoznunk , hogy hol vágjuk el az útvonalat kisebb útszakaszokra.nHa a hálózatnak van egy jó közösségszerkezete, akkor a távolság két azonos közösségben lévő csúcs között sokkal kisebb, mint a távolság két különböző közösségben lévő csúcs között. Először a nagyobb távolságok mentén kellene felosztani a pontokat. Hierarchikus struktúrát kapunk, ha csökkenő sorrendben

**4. Tesztelés**

Ebben a fejezetben az algoritmusok eredményeit hasonlítjuk össze NMI pontosság és futásidő szerint. Az algoritmusok teljesítményeit valós világbeli és mesterségek hálózatokon értékeljük ki. A teszteléshez legfeljebb néhány száz csúcsból álló gráfokat használunk az algoritmusok komplexitása miatt.

A tesztek egy Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz processzorral, 8,00 GB memóriával ellátott számítógépen, Windows 10 operációs rendszeren lettek elvégezve.

**4.1. NMI pontosság**

A klaszterezések kiértékeléséhez az egyik jól ismert teljesítmény mérő az *NMI (normalized mutual information)*. Az NMI az optimális közösségek és az algoritmus által talált közösségek hasonlóságát minősíti. Adott G hálózat optimális közösségeinek halmaza legyen C(A), és az adott algoritmusból kapott közösségeinek halmaza legyen C(B).

ahol n a hálózat pontjainak száma, CA az alapigazság közösségek száma, CB a talált közösségek száma, nij azon i valós alapigazság lévő csúcsok száma, amelyek megtalálhatók a j talált közösségben is , ni az i alapigazság közösségben lévő csúcsok száma, nj a j talált közösségben lévő csúcsok száma.

Az NMI értéke 0 és 1 közötti valós szám, minél jobban hasonlít az adott algoritmus által visszaadott közösségi hozzárendelés az alapigazságra, annál nagyobb. Abban az esetben, ha a talált közösségek azonosak az alapigazsággal, az NMI egy, ha teljesen különböznek, akkor nulla.

A pontosság mérésére elkészítettem Python nyelven a saját implementációmat nmi.py néven. Az nmi(A,B) függvény két paramétert vár, az egyik az alapigazságot, a másik a talált közösségeket tartalmazó DAT fájl. Az input fájlokban a közösségszerkezet él és a hozzárendelt közösség párok listájaként írják le. A program beolvassa a fájlokat, kiszámítja a talált közösségszerkezet NMI értékét, és ezt az értéket adja vissza.

**4.2. Alapigazság gráfok**

Léteznek közösségkereső algoritmusok tesztelésére használt valós életbeli standard teszt gráfok, amelyeknek ismerjük a közösségszerkezetét. A teszt hálózatok megtalálhatók Mark Newman oldalán [] GML formátumban. A gráfokat a NetworkX csomag függvényeivel alakítottam át.

Zachary karate klub [karate] hálózata egy amerikai egyetem karate klubjának barátságait jelöli. Tudjuk, hogy a 34 tagból álló klub egy konfliktus következtében két külön klubra bomlott, az algoritmusok által talált közösségszerkezetet ehhez tudjuk hasonlítani.

Dolphin social network

A Les Miserables hálózat csúcsai a Nyomorultak című regény szereplői. Ha két szereplőt összeköt él, az azt jelenti, hogy a szereplők megjelennek ugyanabban a fejezetben.

College Football gráf [fball] LP 0.6046 labdarúgó mérkőzéseket reprezentál, ahol a csapatok a gráf csúcsainak felelnek meg, és két csapat között akkor helyezkedik el él, ha rendszeresen mérkőztek meg egymással a szezonban. Az alapigazság közösségszerkezet a csapatokat konferenciákra osztja. Általában több meccs kerül megrendezésre konferencián belül, mint konferenciák között.

Books about U.S. politics [books] hálózat pontjai a az amazon.com-ról vásárolt amerikai politikáról szóló könyveket reprezentálják, az élek a gyakran együtt vásárolt könyveket kötik össze. Newman szétosztotta a könyveket liberális és konzervatív könyvek osztályára. A könyvek egy részéről nem tudta egyértelműen eldönteni, melyik ideológiához tartoznak. LP 0.5272

**4.3. Gráfgyár bemutatása (röviden)**

A mesterséges hálózatainkat a gráf generátorral [] állítjuk elő.

A program három fájlt készít. A network.dat a generált hálózat éllistáját tartalmazza soronként, a csúcsokat egész számokkal jelöli 1-től kezdve, növekvő sorrendben. Minden él kétszer szerepel. A community.dat-ban a hálózat csúcsainak listája szerepel, a csúcsokhoz hozzárendelve az őket tartalmazó közösséget. A közösségeket is egész számokkal jelöli, 1-től indítva a számozást. A statistics.dat

paraméterek

**Irodalomjegyzék (legalább 1 oldal OK, mert kell legalább 10 elem (cikk v könyv))**

[] FORTUNATO, S. Community detection in graphs. Physics Reports 486, 75-174 (2010)

az alapfogalmakhoz főleg ezt használtam

[] Newman, M. E. J. The structure and function of complex networks. SIAM Review 45, 167-256 (2003)

kicsit ebből is

<https://www.inf.u-szeged.hu/~london/Halozatok/halozat5.pdf>

<https://www.inf.u-szeged.hu/~london/Halozatok/halozat1.pdf>

a magyar kifejezésekhez kellett, egyébként az első 2 cikket itt találtam

[1] Z. Jiang, J. Liu, S. Wang, Traveling salesman problems with PageRank Distance on complex networks reveal community structure, Physica A (2016)

tsp-s részhez kellett nyilván

[2] S. Brin and L. Page, “The anatomy of a large-scale hypertextual Web search engine,” Computer Networks

and ISDN Systems, 30, 107-117, 1998.

tsp prf

[] ARMAN FERDOWSI, ALIREZA KHANTEYMOOR Discovering Communities in Networks: A Linear Programming Approach Using Max-Min Modularity

ilp modellhez és a közösséges fogalmakhoz

[] G. Agarwal and D. Kempe, Modularity-maximizing graph communities via mathematical programming

ilp modellhez

[] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti, “Column generation algorithms for exact modularity maximization in networks,” Physical Review E, vol. 82, no. 4, p. 046112, 2010

ilp modellhez

[] R. Fourer and D. Gay, The AMPL Book. Pacific Grove: Duxbury Press, 2002

ampl

[] <https://www-personal.umich.edu/~mejn/netdata/> Newman network data:

[karate] W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33, 452-473 (1977)

[football] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002)

teszt gráfok

[] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, “Comparing community structure identification,” Journal of Statistical Mechanics: Theory and Experiment, vol. 2005, no. 09, p. P09008, 2005.

NMI-hez