NeMa: Gyors gráfkeresés címke hasonlóság alapján

Arijit Khan
Charu C. Aggarwal

Yinghui Wu
Xifeng Yan

Computer Science
University of California, Santa Barbara

IBM T. J. Watson Research
Hawthorne, NY

1. Rövid összefoglalás

Egyre gyakrabban találni való életbeli adatokat, amelyek címkézett, heterogén összetevők hálózataként vannak reprezentálva. Ahhoz hogy kereséseket végezzünk ezekben a gráfokban, az embernek gyakran azonosítania kell egy adott keresett gráf illeszkedéseit egy (tipikusan hatalmas) hálózatban ami célgráfként van modellezve. A célgráfban lévő zaj és fixált séma hiánya miatt, a keresett gráf jelentősen különbözhet a célgráfban lévő illeszkedésektől mind struktúrailag mind csúcscímkézést tekintve, így állítva kihívásokat a gráfkeresési feladatok számára. Ebben a cikkben mi a NeMa-t (Network Match) ajánljuk, ami egy szomszédság alapú részgráfillesztő technika valós hálózatokban való keresésre.
(1) Hogy egy illesztés minőségét felmérjük, egy újszerű részgráfillesztési költségmetrikát ajánlunk, ami összegzi az egyedi csúcsok illesztésének költségét és egyesíti a strukturális és a csúcscímkézési hasonlóságokat.
(2) A metrikára alapozva, megfogalmazunk egy minimum költségű részgráfillesztő problémát. Adott keresett gráf és célgráf esetén a probléma a keresett gráf (legjobb-k) minimum költségű illesztésének megtalálása a célgráfban. A cikk megmutatja, hogy a probléma NP nehéz.
(3) A probléma megoldására egy heurisztikus algoritmust ajánlunk egy következtetési modellre alapozva. Valamint optimalizálási technikákat is kínálunk az eljárásunk hatékonyságának javítására.
(4) Empirikusan igazoljuk, hogy NeMa hatékony és eredményes is, összehasonlítva a **kulcsszavas kereséssel és különböző modern gráfkereső technikákkal.**

2. Bevezetés

Az internet eljövetelével az adatok forrása drasztikusan megnőtt, beleértve a világhálót, a szociális hálózatokat, a gén adatbázisokat, tudás gráfokat, orvosi és állami feljegyzéseket. Az ilyen adatok gyakran vannak gráffal reprezentálva, ahol a csúcsok címkézett entitások és az élek a köztük lévő kapcsolatokat fejezik ki [14, 43]. Gráfadatok lekérdezése és bányászása létfontosságú a felmerülő programok széles körében [1, 15, 30].

Ahhoz hogy keressünk ezekben a gráfokban, gyakran szükséges azonosítani az illeszkedéseit egy adott keresett gráfnak egy (tipikusan hatalmas) célgráfban. A tradicionális gráfkereső modellek általában részgráfizomorfizmus és kiterjesztései szempontjából vannak definiálva, amely azonosítja azon részgráfokat amelyek pontosan vagy megközelítőleg izomorfak a keresett gráfokkal [35,33, 43]. Ezen túlmenően, lekérdező modellek és nyelvek széles köre ajánlott – mint például SPARQL és XDD, az RDF és XML adatokhoz – amelyek megkívánják egy általános séma meglétét keresett- és célgráfokhoz. Mindazonáltal, a valós gráfok komplexek, zajosak, és gyakran hiányolják az általánosított sémákat [1]. Valóban, (a) a csúcsok lehetnek heterogének, különböző entitásokra hivatkozók (pl.: személyek, cégek, dokumentumok) [14]. (b) Csúcsok címkéi egy gráfban gyakran gazdag szemantikával rendelkeznek (pl.: id, URL-ek, személyes információk, log-ok, vélemények) [15]. (c) Ami még rosszabb, hogy az entitások szemantikája és a különféle adathalmazokban való összekapcsolódásuk különbözhet és ismeretlen lehet a felhasználók számára [1]. Ebben a kontextusban egy illesztés talán nem szükségszerűen (vagy akár megközelítőleg) izomorf a keresett gráffal a címke és topologikus egyenlőség szempontjából. Így hát a tradicionális gráfkereső technikák nem képesek jó minőségű illesztéseket találni.

Figyeljük meg a következő IMDB mozifilm-adathalmazos példát.

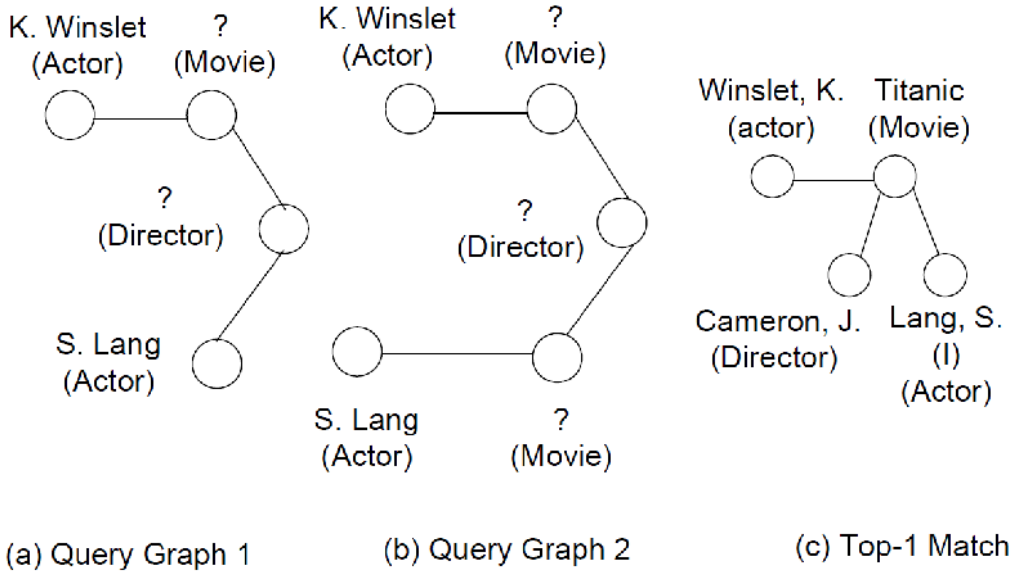


Figure 1: A Query and Its Match (Example 1.1)

Példa 1.1. *A felhasználó egy olyan mozifilmet akar találni, amelynek szereplője 'Kate Winslet' és ugyanaz a rendező rendezte aki egy 'Stephen Lang' által játszott filmet is. Még ha a séma és a pontos címkéi az entitásnak a cél hálózatban nem is érhetőek el, a felhasználó akkor is ki tud találni a kereséshez néhány ésszerű gráfreprezentációt [15, 17], ahogy mutatja a 'Figure 1' nevű kép (a) és (b) része. Megfigyelhető, hogy az ilyen grafikus reprezentációk talán nem lesznek egyediek, és a keresett gráfnak talán nem lesz egy pontos illeszkedése az adathalmazban. Valóban, az eredmény az ábra (c) részén semmikép nem hasonló az ábra (a) és (b) részén látható keresett gráfokhoz a hagyományos gráfhasonlósági definíciók szerint. A célgráf szerkesztési távolsága (annak mértéke,*

hogy hányszor kell gráfszerkesztő operációt végrehajtani ahhoz hogy egyik gráfot a másikká alakítsuk) az 1 és 2 keresett gráfokkal 4 és 6. A legnagyobb közös részgráf mérete 3 mindkét esetben. Akárhogy is, 'Titanic' a helyes válasz a keresésre; és így, az eredménygráf egy jó találatnak tekinthető mindkét keresett gráf számára, valamilyen újszerű gráfhasonlósági metrika használatával.

Ez az, ami a cikk íróit motiválja, hogy lekérdezések megválaszolására alkalmas, gyors részgráfillesztő technikákat vizsgáljanak, amelyek lazíthatnak az részgráfizomorfizmus merev struktúrai és címkeillesztési megkötésein és más hagyományos gráfhasonlósági szabályokon. A cikk írói által ajánlott gráfhasonlósági metrika a következő megfigyelésekre alapszik: (a) ha két csúcs közel van a keresett gráfban, akkor a hozzájuk tartozó eredménygráfbeli csúcsok szintén közel kell hogy legyenek. Mindemellett, (b) lehet néhány különbség az illesztett csúcsok címkéi között.

	NeMa	BLINKS †	IsoRank	SAGA	NESS †	gStore
Precision (Node)	**0.91**	0.52	0.63	0.75	Filter: 0.17 Filter+Verify: 0.80	0.59
Recall (Node)	**0.91**	0.52	0.63	0.75	Filter: 0.83 Filter+Verify: 0.80	0.59
Precision (Graph)	**0.88**	0.50	0.40	0.69	Filter: 0.39 Filter+Verify: 0.74	0.55
Recall (Graph)	**0.88**	0.50	0.40	0.69	Filter: 0.75 Filter+Verify: 0.74	0.55
Top-1 Match Finding Time (sec)	0.97	1.92	4882.0	15.95	Filter: 0.59 Filter+Verify: 56.16	**0.92**

Table 1: NeMa vs. Kulcsszavas Keresés és Gráfkereső Eljárások: A keresett gráfok az IMDB gráfból lettek kivéve és utólag módosítva 30% strukturális és 50% címkézési zaj hozzáadásával. Az írók különböző eljárásokkal meghatározták a legjobb-1 illesztést minden keresett gráfhoz, és felmérték a hatékonyságot az (a) keresett csúcsok és a (b) keresett gráfok szintjén. A csúcsok szintjén a pontosság, avagy precízió (Precission) a helyesen felfedezett csúcsilleszkedések és az összes felfedezett csúcsilleszkedés arányaként van definiálva, míg a visszaidézés (Recall) a helyesen felfedezett gráfillesztések és az összes helyes gráfillesztés arányaként van mérve. Egy gráfillesztést helyesnek feltételezünk, ha legalább 70%-a a csúcsainak helyesen van illesztve. Mivel csak a legjobb-1 illesztést vesszük figyelembe, ezért a precission és a recall értékei megegyeznek. Valamint a NESS szűrő fázis recall és precission értékei is fel vannak tüntetve.

Míg az igény egy gráfhasonlósági metrikára nyilvánvaló (pl.: SAGA[35], IsoRank [33]), kevesen dolgoznak hatalmas hálózatokban való részgráfillesztésen mindkét kritériumot figyelembe véve. Az utóbbi időben a NESS[20] lett kínálva részgráfillesztésre, ami figyelembe veszi a csúcsok közelségét, de egy szigorú csúcscímkeillesztéshez folyamodik. A NESS algoritmus egy szűrés-és-verifikáció megközelítésre alapul. A szűrő fázisban a kevésbé ígéretes csúcsjelöltek ki vannak szortírozva iteratívan, amíg már nincs több kiszortírozható jelölt. A szűrő fázis kimenete egy limitált számú végső jelölt mindegyik keresett csúcs számára. Ezután az algoritmus megerősíti az ezen végső jelöltek által formált összes lehetséges gráfilleszkedést, azért hogy megtalálja a legjobb-k gráfillesztést. A NESS módosítható, hogy alkalmazkodjon csúcscímke különbségek létezéséhez. Azonban ez a módosítás csökkenti a szűrő fázis hatékonyságát, és a végső jelöltek nagy számát eredményezi mindegyik keresett csúcs esetén. Valójában, az írók kísérleteiben, nagyon kevés a precíziós találata a NESS-nek a szűrő fázis végén (lásd Table 1). Ezért elég drágává válik a legjobb-k gráfilleszkedés meghatározása a nagy számú végső jelöltekből. Ezzel szemben az írók által ajánlott NeMa framework egy olyan következtető algoritmust alkalmaz, amely iteratívan növeli a még ígéretesebb csúcsjelöltek találatát, figyelembe véve a címkézésbeli és a strukturális hasonlóságokat is, és így közvetlenül találja meg a legjobb-k gráfilleszkedést.

Amivel a cikk hozzájárul a témához: Ebben a cikkben az írók a NeMa-t ajánlják, egy újszerű részgráfillesztő framework-t, heterogén hálózatokban való kereséshez.
(1) A keresési eredményt az írók egy adott keresett gráfnak a célgráfban való illeszkedéseként definiálják, homomorfizmus alapú részgráfillesztési szándék szempontjából. Az illeszkedések minőségének felmérésére egy újszerű részgráfillesztési költségmetrikát definiálnak a keresett gráf és illeszkedései között. Ellentétben a szigorú részgráfizomorfizmussal, az általuk ajánlott metrika összegzi a különböző keresett csúcsok illeszkedését, ami függ a csúcsok címkéinek illesztésének költségétől és a bizonyos lépésszámra lévő szomszédságuk illesztési költségeitől.
(2) A költségmetrikára alapozva, az írók egy részgráfillesztési minimumköltség-problémát kínálnak, amely szerint a keresett gráf legolcsóbb illesztését keressük a célgráfon belül. Megmutatják, hogy a probléma NP-nehéz.
(3) Az említett problémára az írók egy heurisztikus eljárást kínálnak. Tömören, a NeMa átalakítja az alapul szolgáló gráfhomomorfizmus-problémát egy ekvivalens következtető-problémává grafikus modellekben [29], és így lehetővé teszi a következtető algoritmus alkalmazását az optimális illesztések heurisztikus azonosításához. Eljárásuk elkerüli a költséges részgráfizomorfizmust és gráfszerkesztési távolságok számítását. Ezenkívül indexelési és optimalizációs technikákat is kínálnak eljárásukhoz.
(4) Az írók empirikusan igazolják a NeMa hatékonyságát és eredményességét. Valós hálózatokon végzett kísérleti eredményeik megmutatják, hogy NeMa gyorsan talál jobb minőségű eredményeket, összehasonlítva a kulcsszavas kereséssel (pl.: BLINKS [16]) és különböző gráfkereső technikákkal (pl.: IsoRank [33], SAGA [35], NESS [20], gStore [3]).

3. Kapcsolódó munkák

Részgráf Illesztés. Ullmann visszalépéses eljárását [38], VF2-t[9], és Gyors-indexelést [32] szokás használni részgráf-izomorfizmus ellenőrzésére.

A részgráfkeresési probléma a keresett gráf összes előfordulását azonosítja a célhálózatban. Bioinformatikában, a pontos és közelítő részgráfillesztést is alaposan tanulmányozták. Lásd.: PathBlast [19], SAGA [35], NetAlign [22], IsoRank [33]. Ezek közül a SAGA hasonlít a cikkírók módszeréhez a probléma megfogalmazása szempontjából. Azonban ezek az algoritmusok kisebb biológiai hálózatokat céloznak meg. Nehéz őket hatalmas heterogén hálózatokra alkalmazni.

Pontatlan részgráfok nagy gráfokra való illesztésével kapcsolatban voltak jelentős tanulmányok. Tong et al. [37] a „best-effort” mintaillesztést ajánlotta, amely megpróbálja megtartani a keresett gráf alakját. Ezzel ellentétben, mi az optimális illesztéseket inkább az entitások közelsége szempontjából azonosítjuk, mint a keresett gráf alakja szerint. Tian et al. [36] egy közelítő részgráfillesztő eszközt kínált hatékony indexeléssel, melynek neve TALE. Mongiovi et. al. egy „set-coverbased” illesztőtechnikát mutatott be pontatlan részgráfhoz, SIGMA néven [26]. Mindkét technika az élek találatát használja az illesztések minőségének mérésére; és ezért, nem képesek magukba foglalni az entitások közelségének fogalmát. Vannak más alkotások is pontatlan részgráffal való illesztéssel kapcsolatban. A teljesség igénye nélküli listánk (lásd [13] szemléért) tartalmazza a homomorfizmus alapú részgráfillesztést [12], a hiedelempropagálás alapú hálóigazítást [4], az élszerkesztési távolság alapú részgráfindexelési technikát [41], a milliárd csúccsal rendelkező gráfokban való részgráf illesztést [34], a reguláris kifejezés alapú gráfmintaillesztést [3], a séma [25] és kiegyensúlyozatlan ontológia illesztést [42]. Közülük a homomorfizmus alapú részgráfillesztés [12] áll a legközelebb a módszerünkhöz. Azonban a legjobb-k illesztés helyett, ez utóbbi az összes olyan részgráfról beszámol, amelynél a keresett élek illeszthetők egy bizonyos maximum hosszúságú úthoz és a címkék közti különbségek bizonyos küszöbön belül vannak.

Több munka készült szimuláció és biszimuláció alapú gráfmintaillesztésről (lásd [11], [23]), amelyek a részgráfillesztést mint keresett- és célcsúcsok közti kapcsolatként értelmezik. Hozzájuk hasonlítva Nema szigorúbb, mivel mi az részgráfillesztést mint függvényt értelmezzük a keresett- és célcsúcsok között.

Címke and Fogalom Propagálás. Címkepropagálás széles körben lett használva félig felügyelt tanulásnál, például címkézetlen gráfcsúcsok címkézésénél [31]. Fogalom Propagálás (CP)/Fogalom Vektor (CV) ugyanakkor eredetileg arra lett megfogalmazva, hogy szemantikus hasonlóságokat mérjünk feltételek/fogalmak között egy taxonómiában [21]. Megjegyezzük, hogy a memóriával kapcsolatos terjedéses aktivációs teória [2] az aktivációs propagálás egy hasonló gondolatát használja. CP/CV és a terjedéses aktiváció hatékonyan lett használva [7, 20] -ban közelítő strukturális illesztésre fákban, gráfokban, valamint összekapcsolt hálózatokból való információszerzésre [5]. Ezek azonban csak szigorú címkeillesztést vesznek figyelembe. Viszont csúcscímkék nélküli részgráfillesztés egy nehezebb probléma, mint részgráfillesztés csúcscímkékkel [40]. Ezért ha szigorú csúcscímke egyezés helyett megengedünk közelítő csúcscímkeillesztést (pl.: ahogy mi csináljuk), akkor a kereső-probléma komplexitása jelentősen megnőhet.

Félig-strukturált adatokban való keresés. Lorel és UnQl olyan lekérdező nyelvek, amelyeket félig-strukturált adatokhoz terveztek. Mindkettő címkézett gráfként modellezi az input adatokat, és megengedi a felhasználóknak, hogy lekérdezéseket írjanak anélkül hogy többet tudnának a sémáról. Később, egy a háttérben lévő lekérdezésfeldolgozó rendszer átalakítja a lekérdezéseket hagyományos SQL-é vagy strukturális rekurziós lekérdezésekké, a helyes válasz megszerzése céljából. A lekérdezés újrafogalmazásának ötlete fel lett fedezve mind a relációs és fél-strukturált adatok kontextusában (pl.: [10, 28, 39, 15]). Figyeljük meg, hogy az ilyen lekérdezésújrafogalmazó technikák megszabadítják a felhasználókat a séma megértésének komplexitásától; miközben a háttérben lévő lekérdezésfeldolgozó rendszer továbbra is igényli a rögzített sémát.

Az RDF világában a SPARQL széles körben van használva, mint lekérdezésfeldolgozó nyelv. Mindazonáltal SPARQL-ben írni gyakran túl nehéz, mivel szükséges hozzá a struktúra, a csúcscímkék és típusok pontos ismerete. gStore [43], az első tanulmány amely felvet RDF adatokban egy részgráfillesztés alapú keresést megválaszoló technikát, és lehetővé tesz közelítő csúcscímkeillesztést, de ragaszkodik a szigorú strukturális illesztéshez. Ezzel ellentétben, a mi NeMa framework-ünk megengedi mind a strukturális mind a csúcscímkézési kategóriában a pontatlan illesztést. Munkánk különbözik a gráfokban való kulcsszavas kereséstől [16, 18], mivel lekérdezéseinknek struktúrája is van, nem csak kulcsszavai (csúcscímkék).

4. Alapfogalmak

Célgráfok, keresések és illesztés.

Célgráf. A heterogén hálózatos adathalmazt reprezentáló célgráf definiálható címkézett, irányítatlan gráfként G=(V, E, L), amiben a csúcsok halmaza V, az élek halmaza E, és a címkéző függvény L, és (1) minden V-beli u célcsúcs egy hálózatbeli entitást reprezentál, (2) minden E-beli e él két entitás közti kapcsolatot fejez ki, és (3) L egy függvény amely minden u csúcshoz egy L(u) véges ábécéjű címkét rendel. A gyakorlatban a csúcscímkék reprezentálhatják az entitások tulajdonságait, például név, érték, stb..

Keresett gráf. A keresett gráf Q = (V_Q, E_Q, L_Q) egy irányítatlan, címkézett gráf, amiben V_Q a keresett csúcsok halmaza, E_Q az élek halmaza, és L_Q egy címkéző függvény, ami minden V_Q-beli v keresett csúcshoz rendel egy $L_Q(v)$ véges ábécéjű címkét.

Következőleg egy (összetett) keresett gráfnak a részgráfillesztését definiáljuk nagy célhálózatban.

$Q(V_Q, E_Q, L_Q)$	Keresett gráf
$G(V, E, L)$	Célgráf
$\phi : V_Q \to V$	Részgráfillesztő függvény
Δ_L	Címkekülönbség függvény
$\mathbb{M}(v)$	V csúcs jelölthalmaza
$\mathbb{R}_G(u)$	U csúcs szomszédságvektora
$N_\phi(v, u)$	Szomszédságillesztés költsége u és v között
$F_\phi(v, u)$	Egyéni csúcsillesztési költség u és v között
$C(\phi)$	Részgráfillesztési költségfüggvény

Table 2: Jelölések: Célgráfok; Keresések; Részgráfillesztés

Adott egy célgráf G = (V, E, L) és egy keresett gráf Q = (V_Q, E_Q, L_Q), (1) a V-beli u csúcs egy jelöltje a V_Q-beli v keresett csúcsnak, ha a különbség a címkéik között (értsd L(u) és $L_Q(v)$), egy adott Δ_L címkekülönbség függvény által meghatározva kevesebb vagy egyenlő mint az előre megszabott küszöb. A v csúcs jelölthalmazát M(v)-vel jelöljük. (2) a részgráfillesztés egy 'many-to-one' függvény: $V_Q \to V$, ami minden V_Q-beli v keresett csúcshoz M(v)-beli eredményt ad.

Megjegyzések. (1) A két csúcs címkéi közötti Δ_L címkekülönbség függvény definiálható különféle feltételekkel, mint például a Jaccard-hasonlóság, a szövegszerkesztési távolság, vagy több szofisztikáltabb szemantikus metrika (pl.: ontológia hasonlóság [10]). Ebben a munkában mi Jaccard-hasonlóság mérést használunk a Δ_L megállapítására. (2) A hagyományos részgráfizomorfizmus teszteknél használt szigorú 'one-to-one' feltérképezés (= mapping) helyett, mi egy általánosabb 'many-to-one' részgráfillesztő függvényt veszünk. Valóban két keresett csúcsnak lehet ugyanaz az illesztése [12, 30]. **(3)** A gyakorlatban a cél és keresett gráfokban lévő csúcsok típusokkal ruházhatók fel (például Figure 1 és [15]), ekkor a keresett csúcs csak olyan célcsúcshoz illeszthető amelynek ugyanaz a típusa. Ilyen esetekben a jelölt halmazaink finomításával a részgráfillesztő modellünk könnyedén módosítható, hogy érzékelje a típusmegkötéseket.

Részgráfillesztési költségfüggvény

Több érvényes illesztő függvény is megadható egy adott keresett gráfhoz és célgráfhoz [13]. Ahogy azt már korábban is írtuk, a mi újszerű gráfhasonlósági metrikánknak meg kell őriznie a keresett gráfban lévő csúcspárok közti közelséget, miközben az illesztett csúcsok címkéjének is hasonlónak kell lenniük. Ezt irányelvként véve, bemutatjuk az részgráfillesztési költségfüggvényt a NeMa-ban, mint az illesztés jóságának mérésére alkalmas metrikát. A függvény összeadja a keresett csúcs jelölt csúcsra való illesztésének költségeit, így ragadva meg két csúcs címkéi, és szomszédságstruktúrái közötti különbséget. Elsőként a szomszédságvektor fogalmát vezetjük be.

Szomszédságvektorizálás. Adott egy u csúcs a G célgráfban, i szomszédságát a szomszédságvektorral reprezentáljuk $R_G(u) = \{<u', P_G(u, u')>\}$, ahol u' egy olyan csúcs amely h távolságon belül van az u csúcstól, és $P_G(u, u')$ jelöli a G-ben vett közelségét a két csúcsnak.

$$P_G(u,u') = \begin{cases} \alpha^{d(u,u')} & \text{if } d(u,u') \leq h; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Ebben, d(u,u') a távolság u és u' csúcsok között. A terjedési tényező egy paraméter 0 és 1 között; és h>0 a távolsága (azaz a sugara) a vektorizáció szomszédságának. Az u csúcs szomszédainak vektora magában foglalja a közelségek információját az u csúcstól a h távolságra lévő szomszédcsúcsokig. Gyakran elegendő kis értéket választani h számára (pl.: h=2), mivel két entitás közötti kapcsolat irrelevánssá válik ahogy a szociális távolság köztük nő [6].

A szomszédok vektorára alapozva, most tovább lépünk a keresett csúcs és célcsúcs szomszédainak illesztési költségének modellezésére. Jelöljük a v csúcs h távolságra lévő szomszédcsúcsainak halmazát N(v)-vel. Ha adott a ϕ illesztő függvény, a v és u= $\phi(v)$ közti szomszédillesztési költséget $N_\phi(v,u)$ -val jelöljük és az alábbiként definiáljuk:

$$N_\phi(v,u) = \frac{\sum_{v' \in \mathbb{N}(v)} \Delta_+ \left(P_Q(v,v'), P_G\left(u, \phi(v')\right)\right)}{\sum_{v' \in \mathbb{N}(v)} P_Q(v,v')} \quad (2)$$

where $\Delta_+(x,y)$ is a function defined as

$$\Delta_+(x,y) = \begin{cases} x - y, & \text{if } x > y; \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Érezhetően, az $N_\phi(v,u)$ felméri a v és u szomszédvektorainak az illesztési költségét. Vegyük észre hogy (i) a felhasználó a kereséseit a saját, entitások célgráfban való kapcsolódásáról alkotott, homályos elgondolására alapozva fogalmazza meg. Ezért Δ_+ elkerüli azon esetek büntetését, amikor két csúcs közelebb van a célgráfban, mint a hozzájuk tartozó csúcsok a keresett gráfban.
(ii) Mi normalizáljuk $N_\phi(v,u)$ -t a v szomszédai felett, ami több költséggel jár amikor azonos számú elvételt csúcs fordul elő egy kisebb szomszédságban. Emlékezzünk, hogy feltesszük egy címkekülönbség függvény létezését 0 <= Δ_L <= 1. Most, a különböző csúcsok illesztési költségét az illesztő függvény szerint úgy definiáljuk, hogy vesszük a lineáris kombinációját a címkekülönbség függvénynek és a szomszédságillesztési költségfüggvénynek.

$$F_\phi(v,u) = \lambda \cdot \Delta_L\left(L_Q(v), L(u)\right) + (1-\lambda) \cdot N_\phi(v,u), \quad (4)$$

where $u = \phi(v)$.

Ez a csúcsillesztési költség kombinálja a címkeillesztési költséget és a szomszédságillesztési

költséget a 0 < Δ_L < 1 paraméter által, aminek optimális értéke empirikusan 0,3 és 0,5 között van. Most már készen állunk, hogy definiáljuk az részgráfillesztő költségfüggvényünk.

Adva van egy ϕ illesztés a V_Q-beli v keresett csúcsokról a V-beli $\phi(v)$ célcsúcsokra, ekkor a részgráfillesztési költségfüggvényt így definiáljuk:

$$C(\phi) = \sum_{v \in V_Q} F_\phi(v, \phi(v)) \qquad (5)$$

Érezhetően $C(\phi)$ az illesztési költsége ϕ -nek a Q keresett gráf és a G célgráf között, és a probléma az hogy találjunk egy ϕ illesztő függvényt, amivel $C(\phi)$ minimum értéket vesz fel. Vegyük észre, hogy feltéve hogy Δ_L nem negatív, az $F_\phi(v, \phi(v))$ és így $C(\phi)$ sem negatív, így a legkisebb érték amit $C(\phi)$ felvehet az a nulla.

Költségfüggvény Tulajdonságai

Az részgráfillesztési költségfüggvényünk következő tulajdonságai illusztrálják a részgráfizomorfizmussal való kapcsolatot.

Tulajdonság 1. Ha a Q kereset gráf részgráfizomorfikus (a struktúra és a csúcscímkék egyenlőségének szempontjából) a G célgráfra, akkor létezik minimum költségű illesztő függvény, melyre $C(\phi) = 0$.

Tulajdonság 1 biztosítja, hogy minden olyan ϕ illesztő függvény, amely pontos (izomorfikus) illesztést talál Q-nak, nulla költségű kell hogy legyen. Azonban egy $C(\phi) = 0$ költségű illesztése Q-nak nem feltétlenül izomorfikus Q-ra. Az ilyen illesztésekre „hamis pontos illesztés”-ként hivatkozunk.

Példa 2.1. Vegyünk egy Q keresett gráfot, egy G célgráfot (Figure 2, következő kép), és egy ϕ részgráfillesztő függvényt, ahol $\phi(v_1) = u_1$, $\phi(v_2) = \phi(v_4) = u_2$, and $\phi(v_3) = u_3$. Feltéve, hogy h=1 és $\alpha = 0.5$, a Q szomszédvektorai a következők: $R_Q(v_1) = \{<v_2,0.5>,<v_3,0.5>\}$, $R_Q(v_2)=\{<v_1,0.5>\}$, $R_Q(v_3)=\{<v_1,0.5>, <v_4,0.5>\}$ és $R_Q(v_4) = \{<v_3,0.5>\}$. Hasonlóan G-ben is vannak szomszéd vektoraink: $R_G(u_1) = \{<u_2,0.5>, <u_3,0.5>\}$, $R_G(u_2)=\{<u_1,0.5>,<u_3,0.5>\}$, és $R_G(u_3) = \{<u_1,0.5>, <u_2, 0.5>\}$. Így hát a különböző csúcsok F_ϕ illesztési költsége 0 minden V_Q-beli v-re és a $C(\phi)$ részgráf illesztési költség nulla. Figyeljük meg hogy a ϕ -vel jelölt illesztési költség nem izomorfikus Q-ra.

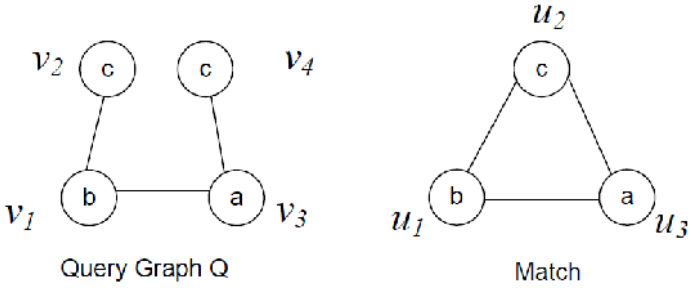


Figure 2: Example of False Exact Match in NeMa

Azonban ha az illesztő függvény 'one-to-one' típusú, akkor a következő tulajdonság megmutatja, hogy a hamis pontos illesztések elkerülhetők.

Tulajdonság 2. Ha a ϕ -vel jelölt illesztés nem izomorfikus a Q keresett gráfra, és ϕ egy 'one-to-one' típusú függvény, akkor $C(\phi) > 0$.

Bizonyítás. Mivel Q összefüggő és ϕ egy 'one-to-one' típusú függvény, ha a ϕ -vel jelölt illesztés nem izomorfikus Q-ra, akkor a következők egyikének teljesülnie kell.

(1) Létezik néhány V_Q-beli v csúcs, feltéve hogy $\Delta_L(L_Q(v), L(\phi(v))) > 0$. Ekkor $C(\phi) > 0$, feltéve hogy λ nem egyenlő nullával a (4). egyenletben.

(2) Létezik E_Q-ban egy (v, v') él, de a hozzá tartozó (u, u') él nincs meg a G gráfban. $\phi(v) = u$ és $\phi(v') = u'$. Ez arra utal, hogy $P_Q(v, v') = \alpha$, de P_G (u, u') < α , amiből aztán következik, hogy $N_\phi(v, u) > 0$. Feltéve hogy λ nem egyenlő 1-gyel a (4). egyenletben, azt kapjuk, hogy $C(\phi) > 0$.
Bizonyítás vége.

5. Eredmények

Probléma megfogalmazása – Minimum költségű részgráfillesztés.
Adott G célgráf, Q keresett gráf, és a címkézési zajküszöb, találj minimum költségű illesztést,

$$\underset{\phi}{\operatorname{argmin}}\, C(\phi), \tag{6}$$

$$s.t. \qquad \Delta_L\left(L_Q(v), L(u)\right) \leq \epsilon, \forall v \in V_Q, u = \phi(v) \tag{7}$$

Érezhető, hogy részgráfizomorfizmus ellenőrzése helyett, a probléma megfogalmazásunk egy optimális illesztést azonosít csúcscímke különbségek és csúcspár távolságok minimalizálásával. A következő állítás szerint a probléma döntési verziója 'intractable', még ha a részgráfillesztő függvény nem is injektív.

Állítás 1. Adott G célhálózat, Q keresett gráf. Annak eldöntése, hogy létezik-e ϕ illesztés NeMa-ban $C(\phi) = 0$ részgráfillesztési költséggel, egy NP-teljes probléma.

A problémát közelíteni is nehéz.
Állítás 2. A minimum költségű részgráfillesztés APX-nehéz.

Keresés feldolgozási Algoritmus

Ebben a részben egy heurisztikus megoldást kínálunk minimum költségű illesztések azonosítására.

Iteratív következtető algoritmus – NemaInfer (Figure 3)

Algorithm NemaInfer

Input: Target graph $G(V, E, L)$, Query Graph $Q(V_Q, E_Q, L_Q)$.
Output: Minimum cost matching of Q in G.

1. **for each** node $v \in V_Q$ **do** compute $\mathbb{M}(v)$;
2. $i := 0$; flag := true;
3. Initiate iterative inferencing with Eq. 8;
4. **while** flag **do**
5. $i := i + 1$;
6. **for each** $v \in V_Q$ **do**
7. **for each** $u \in \mathbb{M}(v)$ **do**
8. compute $U_i(v, u)$ with Theorem 3;
9. keep track of the current matches of neighbors $v' \in \mathbb{N}(v)$;
10. compute optimal match $O_i(v)$ using Eq. 10;
11. if more than a threshold number of query nodes v satisfy $O_i(v) = O_{i-1}(v)$ **then**
12. flag := false;
13. construct Φ for all $v \in V_Q$ (with Eq. 11, 12).
14. **return** Φ;

Figure 3: Iterative Inference Algorithm NemaInfer

Áttekintés. Adott Q keresett gráf, G célgráf, NemaInfer először kiszámítja minden keresett csúcs jelölthalmazát az L csúcscímke hasonlósági függvény használatával (1. sor). Ezután inicializál egy $U_0(v, u)$ következtetési költséget, úgy hogy hozzárendeli az egyes csúcsok $F_\phi(v, u)$ illesztési költségeinek legkisebb lehetséges értékéhez, minden lehetséges ϕ illesztő függvény esetén, feltéve hogy $\phi(v)$ =u (2-3. sor). Aztán iteratívan kiszámítja a következtetési költséget minden keresett v csúcs és jelöltei számára, és minimális következtetési költséggel kiválasztja az optimális illesztését v-nek, ami annak egy u jelöltje. NeMaInfer nyomon követi az összes keresett csúcs optimális illesztését. Az eljárás addig ismétlődik, amíg egy fixponthoz nem ér, azaz több mint adott mennyiségű keresett csúcs optimális illesztése változatlan marad két egymást követő iterációban. (4-12. sor)

Végezetül, NemaInfer pontosítja az illesztését minden keresett csúcsnak és szomszédságának, amit „megjegyez” egy memorizációs technika segítségével, és eléri a legjobb illesztést (13. sor). Végül visszaadja a megkonstruált ϕ részgráfillesztést (14. sor)

A NeMaInfer eljárásairól, valamint indexelésről és optimalizációról részletesebben az eredeti cikkben olvashatsz (5-6. fejezet).

Következtetési költség minden V_Q-beli v és M(v)-beli u-ra az i-edik iterációban:

$$U_0(v, u) = \min_{\{\phi:\phi(v)=u\}} F_\phi(v, u) \tag{8}$$

$$U_i(v, u) = \min_{\{\phi:\phi(v)=u\}} \left[F_\phi(v, u) + \sum_{v' \in \mathbb{N}(v)} U_{i-1}\left(v', u'\right)\right] \tag{9}$$

Optimális illesztése v keresett csúcsnak az i-edik iterációban:

$$O_i(v) = \underset{u \in \mathbb{M}(v)}{\operatorname{argmin}} U_i(v, u); \quad i \geq 0 \tag{10}$$

Példa 4.1.

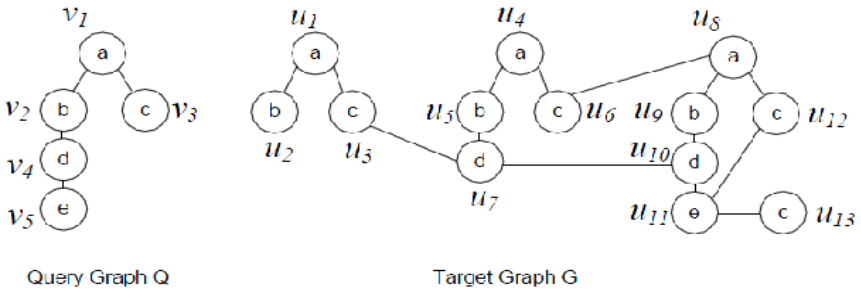


Figure 4: Optimal Subgraph Match Finding Algorithm

A NemaInfer egy iterációjának egy illusztrációja látható az ábrán. Tételezzük fel, hogy már meghatároztuk az M(v) illesztésjelölteket minden v keresett csúcshoz, a Δ_L címke hasonlósági függvény használatával. Például legyen $M(v_2)=\{u_2, u_5, u_9\}$ és $M(v_4) = \{u_7, u_{10}\}$. És tételezzük fel hogy h=1. Az i=0 esetén $U_0(v_2, u_5) = U_0(v_2, u_9) = 0$. Ebből kifolyólag nem tudunk különbséget tenni u_5 és u_9 között az inicializációs körben annak szempontjából, hogy melyik a jobb illesztése v_2-nek. De figyeljük meg hogy $U_0(v_4, u_{10}) < U_0(v_4, u_7)$. Az u_{10} az u_9-nek, míg az u_7 az u_5-nek a szomszédja. Ezért ez nem csak az $O_0(v_4)$ optimális illesztését befolyásolja a v_4-nek az i=0 iterációban, de az i=1 iterációban a (9) egyenlet miatt az is igaz lesz, hogy $U_1(v_2, u_9) < U_1(v_2, u_5)$. Így hát az illesztéseket minden iterációban javítjuk és haladunk a minimum költségű (heurisztikus) részgráf illesztés felé.

Kísérleti eredmények

Bemutatunk három adag empirikus eredményt valós adathalmazokra, hogy kiértékeljük a hatékonyságot és eredményességek, a skálázhatóságot, és az optimalizálási technikákat.

Kísérlet felállítása

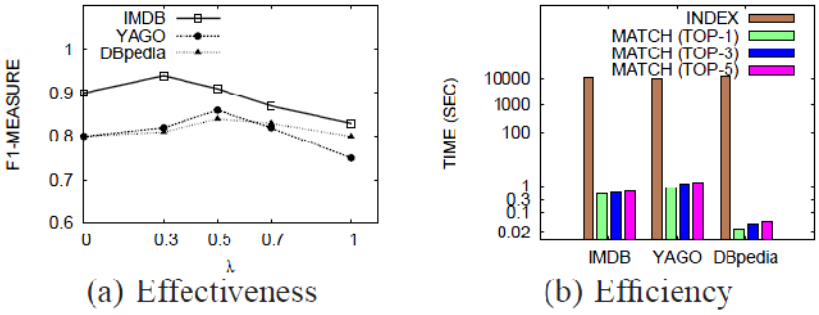
Gráf adathalmazok. A következőkben említett három valós adathalmazt használtuk, mindegyik egy célgráfot reprezentál. (1) **IMDB Hálózat.** Internetes mozi adatbázis, a filmek az entitások az adataikkal és a köztük lévő kapcsolatokkal (pl.: Producer). (2) **YAGO Entitás Kapcsolati Gráf.** YAGO egy tudás adatbázis, amely a Wikipédiából, WordNetből és GeoNamesből meríti az információit. (3) **Dbpedia Tudás Bázis.** Dbpedia a Wikipédiából nyeri az információt.
YAGO és Dbpedia csúcsai címkézettek, míg az IMDB csúcsai típussal is rendelkeznek.

Keresett gráfok. A keresett gráfokat a célgráfokból való részgráfkiválasztással generáltuk, és aztán mindegyikhez adtunk zajt. (Részletek a cikkben.)

Kiértékelési metrikák. Mivel a keresett gráfok a célgráfokból voltak kivéve, a helyes csúcsillesztés alapból ismert. A NeMa effektivitását a következőképen mérjük. Precízió **(P)** nem más, mint a helyesen felfedezett csúcsillesztések és az összes felfedezett csúcsillesztés aránya. **Recall (R)** a helyesen felfedezett csúcsillesztések és minden helyes csúcsillesztés aránya. F1-mérés kombinálja ezen kettő eredményét:

$$F1 = \frac{2}{(1/R + 1/P)} \qquad (17)$$

Eljárások összehasonlítása. Összehasonlítottuk NeMa-t a kulcsszavas kereséssel (BLINKS [16]) és különböző gráfkereső eljárásokkal: SAGA [35], IsoRank [33], NESS [20], és $NeMa_{gs}$ – ami a Nema egy gStore-t [43] követő változata. Az összes eljárás C++ nyelven lett implementálva.



(a) Effectiveness (b) Efficiency

Figure 5: Query Performance

Kísérleteinkben (1) az α propagálási faktor és a h vektorizációs mélység 0.5 és 2 értékekre voltak állítva [20], (2) a különböző adathalmazok számára a λ aránykonstanst (4. egyenlet) empirikusan kaptuk (Figure 5(a)), (3) az indexek merevlemezen voltak tárolva. Minden kísérletet az egy magos, 100GB-os, 2.5GHz Xeon szerveren futtattunk.

Hatékonyság és eredményesség

Valós adathalmazok feletti teljesítmény (nézzük a Figure 5 képet)

Ez a kísérlet három valós gráf felett zajlott, átlagosan több mint 100 kereséssel (Figure 5). Minden célgráfhoz, véletlenszerűen generáltunk 100 keresett gráfot, úgy hogy $|V_Q|$=7 and D_Q = 3 volt. Fixáltuk a strukturális zajt 30%-on, a címkézési zajt 50% és a címkézési zajküszöböt 50%-on.

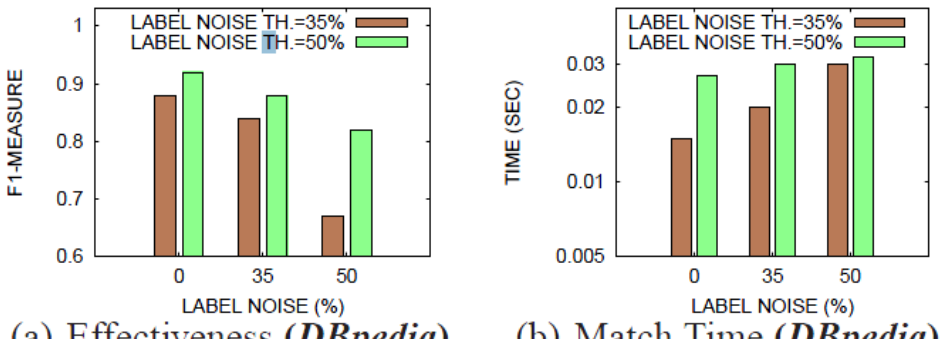
A Figure(a) ábrán látható három adathalmazhoz algoritmusunk mindig helyesen azonosította a keresett csúcsok több mint 76%-át. Specifikusan, az F1-mérés 0,94 -et adott az IMDB-n Y=0.3-ra, még akkor is amikor 30% strukturális és 50% címkézési zajt adtunk hozzá. Az effektivitás magasabb az IMDB-n a benne lévő típusmegkötések miatt. A λ optimális értéke mindegyik adathalmaznál 0.3 és 0.5 között volt.

A Figure(b) a hatékonyságát mutatja az 'off-line' index konstrukciónak (INDEX) és az online keresés kiértékelésnek (MATCH). A következőket figyeltük meg. (a) NeMa kevesebb mint 0.2 másodperc alatt azonosította a legjobb illesztést mind a három adathalmazon. (b) A legjobb-k illesztés megtalálási ideje nem változik jelentősen a k módosításával, mivel a közelítő algoritmusunk mindig csak egyszer fut le. (c) Nem sok idő szükséges az indexeléshez (pl.: 9862 másodperc a YAGO adathalmazon 13M csúccsal és 18M éllel). (d) Az IMDB tömörsége miatt ezen az adathalmazon nagyobb az indexelési és keresési idő.

Hatékonyság vs. Zaj (nézzük a Figure 6 (lenti) képet)

Ezekben a kísérletekben azt vizsgáltuk, hogy a különböző zajok hogyan befolyásolják a NeMa hatékonyságát. Három adag keresett gráf lett generálva, mindegyikben 100 gráffal, a következő beállításokkal: (i) $|V_Q| = 3$, $D_Q = 1$, (ii) $|V_Q| = 5$, $D_Q = 2$, (iii) $|V_Q| = 7$, $D_Q = 3$.

Figure 6 :



(a) Effectiveness (***DBpedia***) (b) Match Time (***DBpedia***)

Különböző címke zajok. A strukturális zajt 30%-on fixálva váltogattuk a címke zajt 0% és 50% között, és így vizsgáltuk a NeMa hatékonyságát, miközben a címkézési zajküszöb 35%-ra és 50%-ra volt állítva. Ahogy a Figure 6(a) képen látható, (1) az F1-mérés értéke csökken, ahogy a címkézési zaj nő, mivel minden keresett csúcs jelölt csúcsai között lehetnek nem „igaz" illesztések, ami így csökkenti a effektivitást (2) az F1-mérés magasabb értéket ad, ha a zajküszöb magasabb, mivel ilyenkor valószínűbb hogy a jelöltek tartalmazzák a helyes illesztéseket. Látható, hogy az F1-mérés mindig 0,60 felett marad.

A Figure 6(b) kép megmutatja, hogy a Nema hatékonysága nem érzékeny a címkézési zajra, viszont érzékenyebb a címke zajküszöbére. Ez azért van, mert a NeMa-nak több időbe telik feldolgozni egy nagyobb jelölthalmazt, ha a küszöb nagyobb.

Különböző strukturális zajok. A címkézési zajküszöböt 35%-on és a címkézési zajt is 35%-on rögzítve, váltogattuk a strukturális zajt 0% és 40% között, ahogy az a Figure 7(a)(b) ábrákon is látható. (Az n_Q jelöli a képeken a keresett csúcsokat, és a D_Q a keresési átmérőt.) Megfigyelhető hogy (a) hatékonyság és eredményesség is csökken ahogy növeljük a strukturális zajt, és (b) a keresés méretének növelésével viszont nőnek az értékek. Ennek oka hogy (1) a nagyobb keresések a keresett csúcs szomszédságára több megkötést tartalmaznak, ami előnyös a helyes illesztések azonosításához, és (2) a NeMa-nak több ideig tart nagyobb keresésekhez az illesztési költségek összehasonlítása. Továbbá, az F1-mérés mindig 0,93 felett van és a futási idő mindig kevesebb mint 0,1 másodperc, még ha a strukturális zaj 40% is.

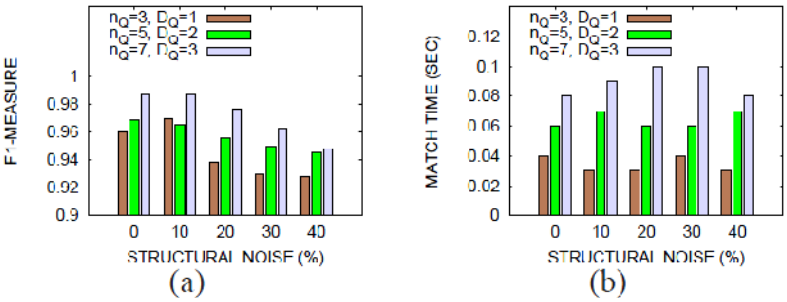


Figure 7: Query Performance vs. Noise (***IMDB***);

Különböző címkézési zajküszöbök. A strukturális zajt 30%-on és a cimkézési zajt 35%-on fixálva, a különböző címkézési zajküszöbök hatását vizsgáltuk a keresési teljesítményre nézve. Az eredményessége és hatékonysága a NeMa-nak az IMDB felett a Figure 7(c)(d) képeken látható. A következőket figyeltük meg. (1) Az F1-mérés kezdettben nő míg növeljük a címkézési zajküszöböt, azért mert a keresett csúcscímkék véletlenszerű szavak hozzáadásával vannak frissítve. Így minél magasabb a címkézési zajküszöb, annál nagyobb az esély, hogy a helyes illesztés lesz kiválasztva a jelölthalmazból. (2) Amikor a címkézési zajküszöb nagyobb mint 35%, az F1-mérés nem változik jelentősen. Így a küszöb optimális értéke empirikusan meghatározható a keresett és célgráfokra alapozva. Másfelől, NeMa futási ideje nő, ha növeljük a küszöböt. Ennek oka (a) az egy keresett csúcsra eső jelöltcsúcsok

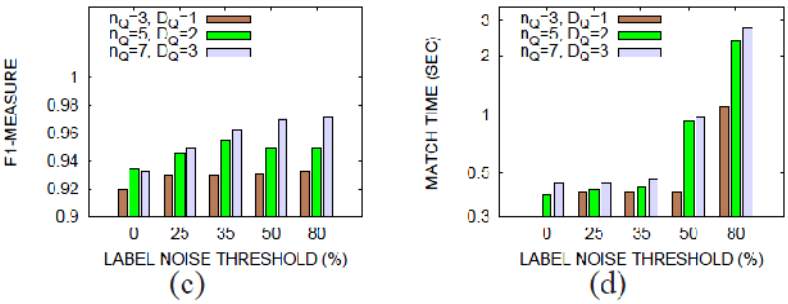


Figure 7: Query Performance vs. Noise (***IMDB***);

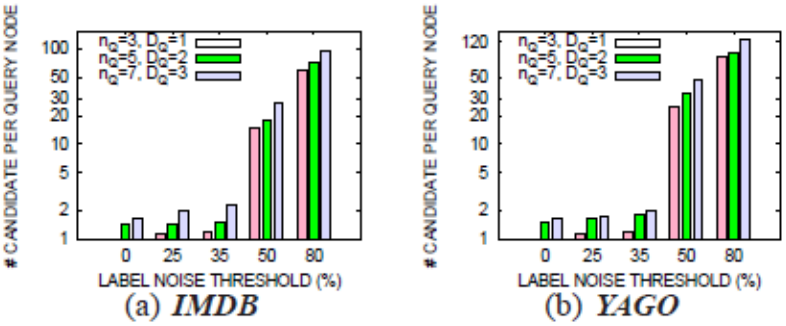


Figure 9: # Candidates vs. Label Noise

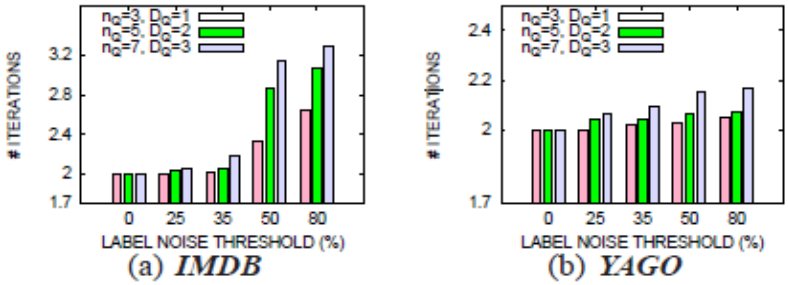


Figure 10: # Iterations vs. Label Noise

számának növekvése (lásd Figure 9), és (b) a hálóillesztő algoritmusunk konvergenciájához szükséges iterációk száma szintén nő (lásd Figure 10), és így NeMa-nak több idő kell az illesztések megtalálására.

Effektivitás címkézetlen keresett csúcsok esetén

Ezekhez a kísérletekhez véletlenszerűen kiválasztottunk két halmazt (mindegyikben 100 keresett gráffal), ahol (i) $|V_Q| = 7$, $D_Q = 3$, és (ii) $|V_Q| = 5$, $D_Q = 2$. A strukturális zajt 30%-on, a címkézési zajt 35%-on, és a címkézési zajküszöböt 35%-on fixálva, váltogattuk a címkézetlen csúcsok számát nullától kettőig. Ahogy a Figure 11 képen látható, (1) az F1-mérés értéke csökken mindkét adathalmaz felett, ha a címkézetlen csúcsok száma nő, mert ezek több jelöltet állítanak, ami így csökkenti az effektivitást, (2) az effektivitás magasabb az IMDB felett a típus megszorítások miatt, és (3) az F1-mérés értéke minden esetben 0.50 felett maradt.

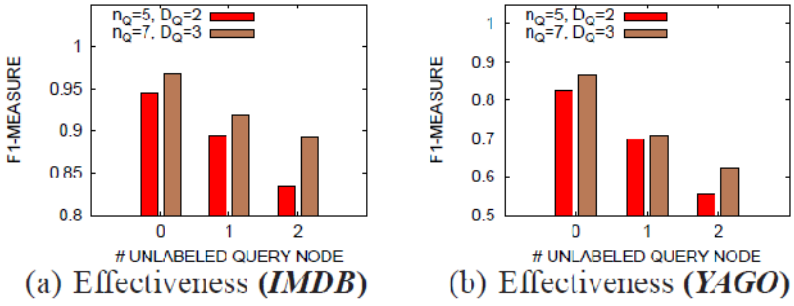


Figure 11: Effectiveness with Unlabeled Query Nodes

Teljesítmény propagálási mélységgel

	$h = 1$	$h = 2$	$h = 3$
Index Time (sec)	265	9862	236553
Match Time (sec)	0.58	0.92	2.76
F1-Measure	0.61	0.86	0.87

Table 4: Query Performance with Varying h ***(YAGO)***

Ezekben a kísérletekben a h propagálási mélység keresési teljesítményre mért hatását analizáltuk. Véletlenszerűen kiválasztottunk 100 keresett gráfot YAGO-ból, úgy hogy $|V_Q| = 7$, $D_Q=3$, a strukturális zaj 30%, a címkézési zaj 50% és a címkézési zajküszöb 50% volt. A Table 4-en látható, hogy a NeMa hatékonysága csökken a h növekedésével, különösen az indexelési idő, ami exponenciálisan nő. Azonban amikor a h = 2 lett, akkor elfogadható 0.86-os értéket kaptunk az F1-mérésre.

Teljesítmény élcímkékkel

Véletlenszerűen kiválasztottuk 100 keresett gráfot az IMDB-ből, úgy hogy $|V_Q| = 7$, $D_Q = 3$, címkézési zaj 50% és címkézési zajküszöb 50%. A strukturális zajt 0% és 40% között választottuk. A keresett gráfban az újonnan beszúrt élek címkéje véletlenszerű string generálással voltak létrehozva. A Figure 12 képen látható, hogy (1) amikor nincs strukturális zaj, akkor az F1-mérés nem változik se címkézett, se a címkézetlen élek esetében. (2) Azonban hozzáadott strukturális zajjal, az F1-mérés értéke kis mértékben csökkent címkézett élek esetén, mivel az új élek címkéi miatt több a zaj a keresett gráfban. (3) A címkézett élek esetében a futási idő nagyobb, mivel idő kell a él címkék hasonlóságának kiértékeléséhez.

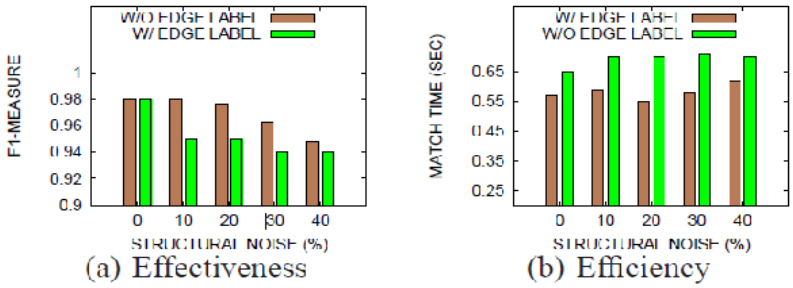


Figure 12: Performance with Edge Labels (***IMDB***)

Létező algoritmusokkal való összehasonlítás

Összehasonlítottuk a NeMa-t az alábbiakkal: IsoRank [33], SAGA [35], NESS [20], gStore [43], BLINKS [16]. (1) IsoRank és SAGA optimális gráfillesztéseket talál kisebb biologikai hálózatokban a strukturális és csúcs címkézési hasonlóságok figyelembe vételével. (2) NESS megtalálja a legjobb-k gráfillesztést egy hatalmas hálózatban, de szigorú csúcscímke egyenlőség vizsgálatával. Emiatt módosítottuk a NESS-t, úgy hogy engedélyezett legyen két csúcs illesztése, ha a címkéik különbsége a címkézési zajküszöbön belül található. (3) Különböző NeMa variációkra is gondoltunk, nevezetesen $NeMa_{gs}$-re, ami engedélyezi a címkék közti különbséget, de szigorú izomorfikus illesztést használ. Ebből kifolyólag $NeMa_{gs}$ lényegében ugyanazon az alapon működik mint gStore, ami egy részgráfizomorfizmus alapú SPARQL keresés kiértékelő eljárás, amiben engedélyezve van csúcs címke különbség. (4) BLINKS [16] egy kulcsszavas keresési eljárás, ami csak a strukturális különbségeket támogatja. Ezért a BLINKS-et is módosítottuk, hogy engedélyezze a csúcs címkék béli különbségeket a címkézési zajküszöbön belül.

Az összes eljárás a NESS kivételével közvetlenül megtalálja a legjobb-1 gráfillesztést. Ezért vettünk egy legjobb-1 illesztést minden keresett csúcs számára, hogy kiértékeljük a precíziót (p), a visszaidézést (R = recall, annak képessége hogy megtaláljuk a lehető legtöbb helyes illesztést), és F1-mérés értékeit; és így ugyanazt a pontszámot értük el számukra. Ezzel szemben a NESS egy szűrés-és-verifikáció megközelítést alkalmaz, ahol a szűrő fázis egy jó minőségű végső csúcsjelölt halmazt állít elő minden keresett csúcs számára. Aztán verifikálja az összes lehetséges gráfillesztést amit a végső jelölt csúcsokból kaphatunk, hogy így találja meg a legjobb-1 gráfillesztést. Ezért a szűrő fázis precízióját (P), visszaidézését (R), és F1-értékelését néztük NESS-nél, mert ez a fázis a legfontosabb lépés ott. Hogy igazságosak legyünk a Figure 13(c) képen, csak a futási idejét vizsgáltuk a NESS-nek.

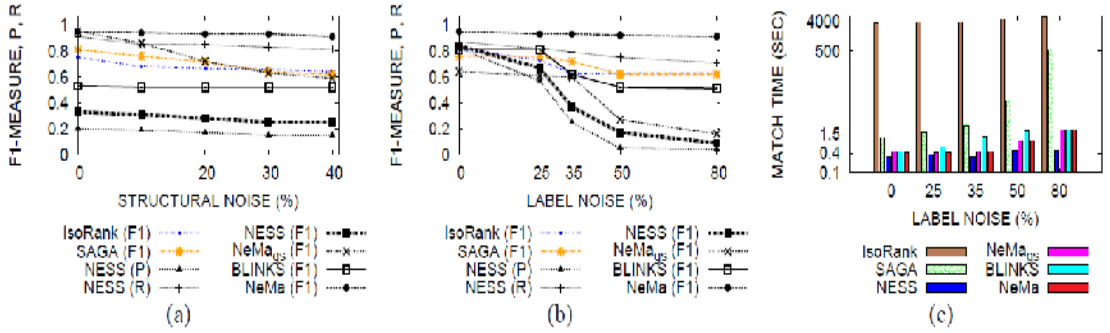


Figure 13: Comparison Results (**IMDB**): NESS, BLINKS Modified for Approximate Label Match. NESS Results Correspond to its *Filtering* Phase.

Ezekhez a kísérletekhez véletlenszerűen kiválasztottunk 100 keresett gráfot az IMDB adathalmazból, úgy hogy $|V_Q| = 7$ és $D_Q = 3$. Minden keresett gráfban volt egy címkézetlen csúcs, és a megmaradt csúcsok címkéi frissítve voltak új szavak véletlen beillesztésével. A strukturális zajt váltogattuk a Figure 13(a) képen, és mind a címkézési zajt, mind a címkézési zajküszöböt 50%-on fixáltuk. Figyeljük meg, hogy strukturális zaj nélkül a NeMA és a $Nema_{gs}$ F1-mérési értéke is 0,94 körül van, de a strukturális zaj növelésével NeMa (F1-érték 0,9) túlszárnyalja a többi eljárást (F1-érték 0,1 ~ 0,7)

A Figure 13(b)(c) képeken váltogattuk a címkézési zajt és a címkézési zajküszöböt, és fixáltuk a strukturális zajt 30%-on. A címkézési zajt és zajküszöböt mindig azonos értékűnek vettük ezeken az ábrákon. Címkézési zaj nélkül NeMa F1-értéke 0,93 volt, míg NESS, IsoRank, SAGA, és BLINKS F1-értéke 0,8 körül. Azonban a címkézési zaj növelésével NeMa (F1-érték 0,9) sokkal nagyobb mértékben túlteljesíti a többi eljárást (F1-érték 0,1 ~ 0,7).

Nema megtalálja a legjobb illesztést kevesebb mint 1 másodperc alatt, míg IsoRank 5000 mp alatt. SAGA-nak 15 mp és 546 mp kell, ha a címkézési zaj 50% és 80%.

Végezetül az eredeti cikkben említett optimalizálási technikákat is megvizsgáltuk, hogy azok hogyan befolyásolják a NeMa hatékonyságát. Javítanak az eredményen. (Részletek: cikk 191.oldal)

Konklúzió

Ebben a cikkben bemutattuk a NeMa-t, ami egy újszerű részgráfillesztés általi gráfkereső algoritmus, ami megengedi, hogy bizonytalanság legyen a struktúrában és a csúcs címkézésben. Egy csúcs szomszédságát többdimenziós vektorrá alakítjuk át, és alkalmazzuk rá a közelítő algoritmust, hogy azonosítsuk az optimális gráfillesztéseket. Továbbá megvizsgáltuk hogyan lehet a NeMa-t kiterjeszteni különböző gráfkeresést feldolgozó alkalmazásra, mint az RDF keresés megválaszolás, a gráfillesztés élcímkék esetén, a legjobb-k közelítő illesztés megtalálása. A kísérleti eredményeink megmutatják, hogy a NeMA hatékonyan talál magas minőségű illesztéseket összehasonlítva a modern gráfkereső eljárásokkal.

6. További kutatás terv

(1) A NeMa célja hogy valós adathalmazokban segítse a keresést. Azonban az ilyen keresések gyakran csak linkeket adnak vissza és nem a kérdést válaszolják meg. Ezért a jövőben érdemes megvizsgálni szofisztikáltabb címkézési hasonlóság metrikákat (például: ontológia és nyelvtani hasonlóságok).

(2) A programok eljárásokból állnak, és ezek sorozatára gondolhatunk gráfként. A kártékony programmintákból előállíthatnánk a keresett gráfok halmazát, és ekkor a vizsgált probléma lenne a célgráf, amelyben NeMa így vírusokat és egyéb kártékony programokat kereshetne.

(3) A jövőben lehet dolgozni az eljárás párhuzamosításán is. Minden keresett csúcsot vizsgálhatunk egy külön szálon, ezáltal tovább gyorsítva a NeMa általi keresést.

(4) Egy másik fejlesztési lehetőség a keresési interface barátságosabbá tétele. Egy felhasználóbarátabb interface, amely nem bemeneti keresett gráfokat kér, elterjedtebbé tehetné az algoritmus használatát. Lehetővé tehetné hétköznapi emberek számára a NeMa használatát.

7. Irodalomjegyzék

[1] S. Abiteboul. Querying Semi-Structured Data. *ICDT*, 1997.
[2] J. R. Anderson. A Spreading Activation Theory of Memory. *J. Verbal Learning and Verbal Behavior*, 1983.
[3] P. Barcel´o, L. Libkin, and J. L. Reutter. Querying Graph Patterns. *PODS*, 2011.
[4] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang. Algorithms for Large, Sparse Network Alignment Problems. *ICDM*, 2009.
[5] H. Berger, M. Dittenbach, and D. Merkl. An Adaptive Information Retrieval System based on Associative Networks. *APCCM*, 2004.
[6] N. Buchan and R. Croson. The Boundaries of Trust: Own and Others' Actions in US and China. *J. Econ. Behav. and Org.*, 2004.
[7] V. S. Cherukuri and K. S. Candan. Propagation-Vectors for Trees (PVT): Concise yet Effective Summaries for Hierarchical Data and Trees. *LSDS-IR*, 2008.
[8] S. Cook. The Complexity of Theorem Proving Procedures. *STOC*, 1971.
[9] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Tran. Pattern Anal. and Machine Int.*, 2004.

[10] S. Das, E. I. Chong, G. Eadon, and J. Srinivasan. Supporting Ontology-Based Semantic Matching in RDBMS. *VLDB*, 2004.
[11] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu. Graph Pattern Matching: From Intractable to Polynomial Time. *PVLDB*, 2010.
[12] W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu. Graph Homomorphism Revisited for Graph Matching. *PVLDB*, 2010.
[13] B. Gallagher. Matching Structure and Semantics: A Survey on Graph-Based Pattern Matching. *AAAI FS.*, 2006.
[14] J. Han. Mining Heterogeneous Information Networks by Exploring the Power of Links. *ALT*, 2009.
[15] L. Han, T. Finin, and A. Joshi. GoRelations: An Intuitive Query System for DBpedia. *LNCS*, 2011.
[16] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: Ranked Keyword Searches on Graphs. *SIGMOD*, 2007.
[17] J. Liu and X. Dong and A. Halevy. Answering Structured Queries on Unstructured Data. *WebDB*, 2006.
[18] M. Kargar and A. An. Keyword Search in Graphs: Finding r-cliques. *PVLDB*, 2011.
[19] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. PathBLAST: A Tool for Alignment of Protein Interaction Networks. *Nucleic Acids Res*, 2004.
[20] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood Based Fast Graph Search in Large Networks. *SIGMOD*, 2011.
[21] J. W. Kim and K. S. Candan. CP/CV: Concept Similarity Mining without Frequency Information from Domain Describing Taxonomies. *CIKM*, 2006.
[22] Z. Liang, M. Xu, M. Teng, and L. Niu. NetAlign: A Web-based Tool for Comparison of Protein Interaction Networks. *Bioinfo.*, 2006.
[23] S. Ma, Y. Cao, W. Fan, J. Huai, and T.Wo. Capturing Topology in Graph Pattern Matching. *PVLDB*, 2012.
[24] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. PREGEL: A System for Large-Scale Graph Processing. *SIGMOD*, 2010.
[25] S. Melnik, H. G.-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. *ICDE*, 2002.
[26] M. Mongiov`ı, R. D. Natale, R. Giugno, A. Pulvirenti, A. Ferro, and R. Sharan. SIGMA: A Set-Cover-Based Inexact Graph Matching Algorithm. *J. Bioinfo. and Comp. Bio.*, 2010.
[27] C. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *J. Comp. and Sys. Sc.*, 1991.
[28] Y. Papakonstantinou and V. Vassalos. Query Rewriting for Semistructured Data. *SIGMOD*, 1999.
[29] J. Pearl. Reverend Bayes on inference engines: A distributed Hierarchical Approach. *American Association of AI Conf.*, 1982.
[30] K. Sambhoos, R. Nagi, M. Sudit, and A. Stotz. Enhancements to High Level Data Fusion using Graph Matching and State Space Search. *Inf. Fusion*, 2010.
[31] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 2008.
[32] H. Shang, Y. Zhang, X. Lin, and J. Yu. Taming Verification Hardness: An Efficient Algorithm for Testing Subgraph Isomorphism. *PVLDB*, 2008.
[33] R. Singh, J. Xu, and B. Berger. Global Alignment of Multiple Protein Interaction Networks with Application to Functional Orthology Detection. *PNAS*, 2008.
[34] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li. Efficient Subgraph Matching on Billion Node Graphs. *PVLDB*, 2012.

[35] Y. Tian, R. McEachin, C. Santos, D. States, and J. Patel. SAGA: A Subgraph Matching Tool for Biological Graphs. *Bioinfo.*, 2006.
[36] Y. Tian and J. M. Patel. TALE: A Tool for Approximate Large Graph Matching. *ICDE*, 2008.
[37] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast Best-Effort Pattern Matching in Large Attributed Graphs. *KDD*, 2007.
[38] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 1976.
[39] J. Yao, B. Cui, L. Hua, and Y. Huang. Keyword Query Reformulation on Structured Data. *ICDE*, 2012.
[40] S. Zampelli, Y. Deville, C. Solnon, S. Sorlin, and P. Dupont. Filtering for Subgraph Isomorphism. *CP*, 2007.
[41] S. Zhang, J. Yang, and W. Jin. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs. *PVLDB*, 2010.
[42] Q. Zhong, H. Li, J. Li, G. Xie, J. Tang, L. Zhou, and Y. Pan. A Gauss Function Based Approach for Unbalanced Ontology Matching. *SIGMOD*, 2009.
[43] L. Zou, J. Mo, L. Chen, M. T. ¨Ozsu, and D. Zhao. gStore: Answering SPARQL Queries via Subgraph Matching. *VLDB*, 2011.