Eötvös Loránd Tudományegyetem
Természettudományi Kar

Gráf- és részgráf-izomorfizmus problémák kémiai adatbázisokban

Diplomamunka

Témavezető:
Dr. Tichler Krisztián
ELTE IK
Algoritmusok és Alkalmazásaik Tanszék
adjunktus

Készítette:
Vigula Mónika
ELTE TTK
Alkalmazott matematikus MSc

Budapest, 2015

Köszönetnyilvánítás

Köszönöm témavezetőmnek, dr. Tichler Krisztiánnak, hogy szakmai és stilisztikai tanácsaival, észrevételeivel segített a diplomamunka megírásában. Köszönöm továbbá, hogy a korábbi szakdolgozatok és a TDK dolgozat készítése során is mindig számíthattam rá.

Köszönöm korábbi témavezetőimnek, dr. Fekete Istvánnak és Kovács Péternek, hogy bevezettek a kémiai informatika érdekes, kihívásokat rejtő világába, és útmutatásaikkal, ötleteikkel és megjegyzéseikkel segítettek a szakdolgozatok és a TDK dolgozat írása közben.

Köszönöm kollégáimnak, hogy visszajelzéseikkel, észrevételeikkel lehetővé tették a folyamatos fejlődést a kémiai informatika területén. Hálával tartozom nekik, amiért elfogadták, hogy egyetemi tanulmányaim miatt nem tudtam minden megbeszélésen részt venni.

Végül, de nem utolsó sorban köszönetet szeretnék mondani Családomnak és Barátaimnak a tanulmányaim során nyújtott biztatásért, támogatásért.

MarvinSketch, felhasználása: kémiai struktúrák rajzolása, Verzió: 15.3.16, 2015, ChemAxon (`http://www.chemaxon.com`)

JChem Base, felhasználása: kémiai struktúrák keresése, adatbázis-kezelés, Verzió: 15.3.16, 2015, ChemAxon (`http://www.chemaxon.com`)

Tartalomjegyzék

Bevezetés

„My big thesis is that although the world looks messy and chaotic, if you translate it into the world of numbers and shapes, patterns emerge and you start to understand why things are the way they are."

Marcus du Sautoy

A gráfok, mint adatmodellek a természettudományos kutatások során széles körben alkalmazhatóak pl. a gyógyszerkutatás, a számítógépes tanítás és a képfelismerés terén. Amennyiben az egyes objektumok reprezentálására gráfot használunk, a feladatot gyakran visszavezethetjük valamely klasszikus gráfelméleti feladatra.

Vizsgálhatjuk két adott gráf hasonlóságát, melyet definiálhatunk – a felhasználási területtől függően – ezen két gráf MCS-ének (Maximum Common Subgraph) vagy MCES-ének (Maximum Common Edge Subgraph) segítségével. Két adott gráf esetén az MCS és az MCES gráf meghatározása is NP-teljes feladat [12], hiszen a dolgozatban tárgyalt részgráf-izomorfia probléma általánosításának tekinthető. Ezen problémát megoldó algoritmusok találhatóak pl. [5]-ben, [7]-ben, [11]-ben, [15]-ben és [22]-ben.

Számos esetben felbukkanó feladat annak eldöntése, hogy két adott gráf izomorf-e. Ezen probléma NP-beli, azonban nem ismert, hogy P-beli vagy NP-teljes-e. Néhány gráftípus esetén létezik polinomiális idejű algoritmus, pl. fokszámkorlátos gráfokra [19], síkbarajzolható gráfokra [14], permutációgráfokra [6] és intervallumgráfokra [18]. A conauto [21] és a nauty [20] gyakran hivatkozott gráfizomorfiát vizsgáló programcsomag.

Klasszikus feladat két adott gráf, G_q (query gráf vagy pattern gráf) és G_t (target gráf) esetén annak eldöntése, hogy G_t tartalmazza-e részgráfként G_q-t, mely szintén NP-teljes probléma. Ezen esetben – a legnagyobb közös részgráf feladathoz hasonlóan – tekinthetünk feszített vagy nem feltétlen feszített részgráfként történő tartalmazást is. Részgráf-izomorfiát eldöntő, visszalépéses keresésen alapuló eljárások

a J. R. Ullmann által publikált algoritmus [29] és az L. P. Cordella és mtsai algoritmusa, a VF2 [8]. Szintén részgráf-izomorfiát vizsgáló algoritmus a QuickSI [24], mely úgy csökkenti a várható futásidőt, hogy kezdőlépésében a címkézett query gráf csúcsait átrendezi az adatbázisbeli csúcs- és élcímkék gyakoriságát felhasználva. A megoldandó gyakorlati feladattól függően elegendő lehet csupán egyetlen részgráf-izomorfizmus megkeresése, de szükség lehet az összes leképezés megtalálására is. Ezen módszerek némi módosítást követően mindkét esetben alkalmazhatóak.

A kémiai informatika területén gyakori feladat, hogy egy adott query molekula és egy target molekulákat tartalmazó adatbázis esetén határozzuk meg azon adatbázisbeli molekulákat, melyek tartalmazzák részstruktúraként az adott query molekulát. Első lépésben a molekulákat címkézett gráfokként ábrázoljuk, majd a feladatot visszavezetjük a páronkénti részgráf-izomorfia vizsgálatra. Tekintettel arra, hogy a gyakorlati alkalmazások során az adatbázisok jellemzően több százezer vagy millió struktúrát tartalmaznak, ezért a páronkénti, költséges részgráf-izomorfia vizsgálat a gyakorlatban kivárhatatlan. A keresési idő csökkentése érdekében az adatbázison előszűrést végzünk, kiszűrve minél több olyan target gráfot, melyek biztosan nem tartalmazhatják részstruktúraként a query molekulát. Az előszűrési lépés során az egyes módszerek általában ún. „fingerprinteket" alkalmaznak, melyek a molekula struktúráját írják le, majd az előszűrést a query és a target molekula fingerprintjét összehasonlítva végzik. A szakirodalomban számos előszűrési módszer található [13, 16, 17, 24, 37, 32, 38].

A kémiai informatikával és a részgráf-izomorfia problémával 2010 őszén kezdtem el foglalkozni. Az ELTE-TTK Matematika BSc, Alkalmazott matematika szakirányán készített szakdolgozatomban [32] az Ullmann algoritmus, a VF2 és a QuickSI algoritmus valamint néhány előszűrési módszer elméletét vizsgáltam. Az ELTE-IK Programtervező informatikus BSc szakon készített szakdolgozatom [30] során a részgráf-izomorfiát eldöntő algoritmusok heurisztikákkal kiegészített implementációjával és a kapott eredmények összehasonlításával foglalkoztam. A szakdolgozatok megvédését követően tovább dolgoztam az algoritmusok futásidejének csökkentése érdekében, az újonnan elért eredményeket egy TDK dolgozat tartalmazza [31], melyeket a MATCOS-13[1] konferencia keretében is prezentáltam [33]. Jelen diplomamunka célja a részgráf-izomorfia probléma CSP[2] feladatként történő megfogalmazása, az ezen megoldó módszerek áttekintése és a korábbi algoritmusokkal történő összehasonlítása elméleti szempontból.

[1] Middle-European Conference on Applied Theoretical Computer Science (MATCOS), Koper, Slovenia, October 10th and 11th, 2013., `http://matcos.iam.upr.si`

[2] Constraint Satisfaction Problem

A molekulák kémiai értelemben vett részgráf-izomorfiája nem egyezik meg pontosan a nekik megfeleltetett címkézett gráfok részgráf-izomorfiájával. A dolgozat során nem célunk a különböző kémiai jelenségek (pl. aromás kötések, tautomerizáció) kezelése, csupán az egyes algoritmusok összehasonlítása elméleti szempontból.

A dolgozat felépítése a következő. Az 1. fejezetben a probléma megértéséhez szükséges alapfogalmakat ismertetjük. A 2. fejezet a visszalépéses keresésen alapuló módszereket foglalja össze. A részgráf-izomorfia feladat CSP feladatként történő megfogalmazását, a lehetséges megoldási módszereket és a korábbi algoritmusokkal történő összehasonlítását a 3. fejezetben tárgyaljuk. A kapott eredményeket a 4. fejezetben foglaljuk össze.

1. fejezet

Elméleti háttér

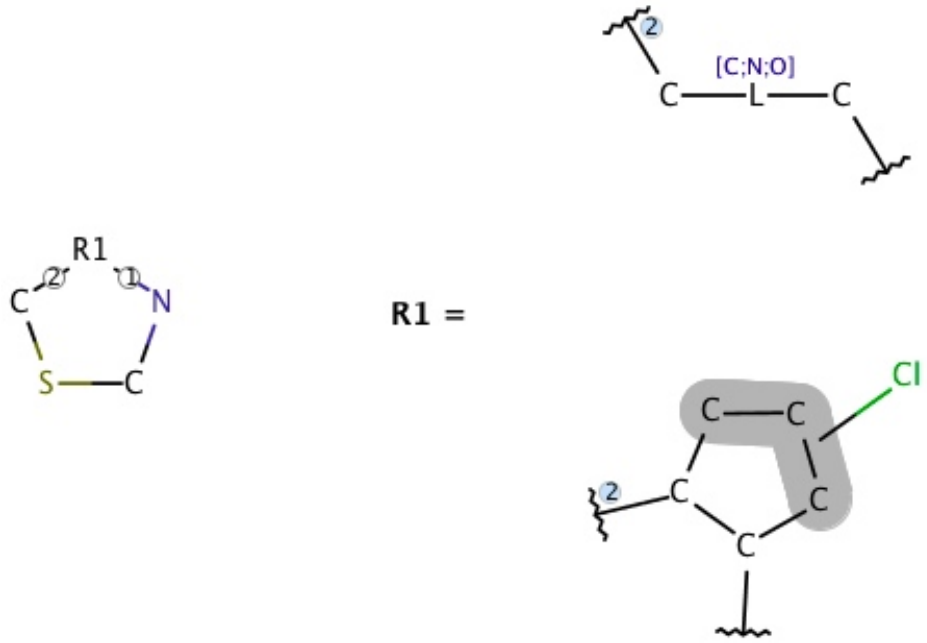
A kémiai informatikában gyakori megoldandó feladat, hogy egy adott lekérdező molekula (query gráf) és egy molekulákat tartalmazó adatbázis esetén határozzuk meg azon adatbázisbeli target molekulákat, melyek részstruktúraként tartalmazzák a query molekulát. Jelen fejezetben ismertetjük, hogy a feladatot miképpen modellezhetjük matematikai eszközökkel: reprezentáljuk a molekulát címkézett gráfként, definiáljuk a részgráf-izomorfia problémát először egy query és egy target molekula esetén, majd egy query és egy molekulákat tartalmazó adatbázis esetén, végül pedig az adatbázison történő előszűréssel foglalkozunk.

1.1. A részgráf-izomorfia probléma

1.1. Definíció *Csúcs- és élcímkézett gráf (a továbbiakban: címkézett gráf) alatt egy* $G = (V, E, \Sigma_V, \Sigma_E, l_V, l_E)$ *hatost értünk, ahol* V *és* E *a gráf csúcs-, illetve élhalmazát jelöli,* Σ_V *illetve* Σ_E *a csúcsok és az élek címkéinek halmaza,* $l_V : V \to \Sigma_V$ *és* $l_E : E \to \Sigma_E$ *pedig a gráf címkézőfüggvényei.*

Egy egyszerű molekulát a következőképpen ábrázolhatunk címkézett gráfként: a gráf csúcsai és élei a molekula egy-egy atomjának illetve kötésének felelnek meg, a csúcs- és élcímkéket pedig a molekula atom illetve kötéstípusainak megfelelően reprezentáljuk. A molekulák rajzolása során általában nem tüntetjük fel a hidrogénatomokat és a rájuk illeszkedő kötéseket, hiszen ezen tulajdonságokat kémiai ismereteink alapján meghatározhatjuk [25]. A dolgozat további részében a molekulák ilyen reprezentálását tekintjük. Megjegyezzük továbbá, hogy a dolgozatban kizárólag olyan molekulákkal foglalkozunk, melyek pontosan egy molekulát írnak le; a Markush-struktúrákat [1] – melyek esetén egy molekula egy több molekulából álló könyvtárat reprezentál – nem tárgyaljuk. Egy tipikus Markush-struktúra tekinthető

meg az 1.1 ábrán[1].



1.1. ábra: Példa a Markush-struktúrára

1.2. Definíció *Legyen adott két címkézett gráf,* $G_q = (V_q, E_q, \Sigma_{V_q}, \Sigma_{E_q}, l_{V_q}, l_{E_q})$ *(query gráf) és* $G_t = (V_t, E_t, \Sigma_{V_t}, \Sigma_{E_t}, l_{V_t}, l_{E_t})$ *(target gráf), melyekre* $V_q \cap V_t = \emptyset$. G_q *részgráf-izomorf* G_t*-vel, ha létezik olyan* $f : V_q \to V_t$ *injektív függvény, melyre az alábbi feltételek teljesülnek:*

1. $\forall u \in V_q \left(l_{V_q}(u) = l_{V_t}(f(u))\right)$
2. $\forall u \in V_q \, \forall u' \in V_q \, ((u, u') \in E_q \Rightarrow (f(u), f(u')) \in E_t)$
3. $\forall u \in V_q \, \forall u' \in V_q \, ((u, u') \in E_q \Rightarrow l_{E_t}((u, v)) = l_{E_t}((f(u), f(u'))))$

Jelölés: $G_q \subseteq G_t$

A definícióban szereplő f függvényt részgráf-izomorfizmusnak nevezzük. A szakirodalomban query gráf helyett találkozhatunk a „pattern gráf” elnevezéssel is. A kémiai informatika területén egy illetve az összes ilyen f függvény meghatározása is releváns.

A fenti problémára a szakirodalomban található egy szigorúbb, feszített részgráfként történő tartalmazást előíró definíció is. Tekintettel arra, hogy a dolgozat

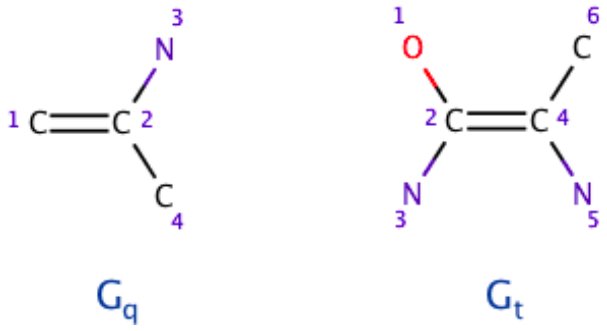
[1] A Markush struktúra mindenképpen tartalmazza az ábra bal oldalán látható molekulát, ahol az $R1$ „atom” helyére az $R1$ valamely definíciója kerül, melyeket az ábra jobb oldalán láthatunk. A definíció helyettesítése során fontos megkülönböztetni az egyes csatlakozási helyeket, melyet a második kapcsolódási pont esetén ② jelöl.

megírásában a kémiai területről érkező kérdések inspiráltak, ezért a továbbiakban a fenti definíciónak megfelelő tartalmazással foglalkozunk. A bemutatott módszerek jelentős része – kis mértékű módosítást követően – alkalmazható a szigorúbb eset vizsgálatára is.

1.1. Példa

Tekintsük az 1.2 ábrán látható query és target gráfot. Egy lehetséges $f : V_q \to V_t$ *:*

$$f(1) = 2 \qquad f(3) = 5$$
$$f(2) = 4 \qquad f(4) = 6$$



1.2. ábra: Példa a részgráf-izomorfia problémára

1.3. Tétel *Legyen adott két címkézett gráf,* $G_q = (V_q, E_q, \Sigma_{V_q}, \Sigma_{E_q}, l_{V_q}, l_{E_q})$ *és* $G_t = (V_t, E_t, \Sigma_{V_t}, \Sigma_{E_t}, l_{V_t}, l_{E_t})$*. Annak eldöntése, hogy* $G_q \subseteq G_t$*, NP-teljes feladat.*

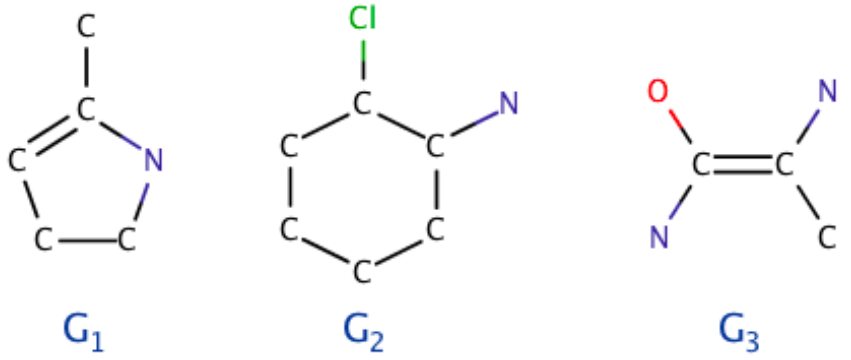
Bizonyítás. A részgráf-izomorfia probléma NP-beli, hiszen polinomiális időben ellenőrizhető tanú az f leképezés, továbbá a problémát probléma könnyen, polinomiálisan visszavezethető az NP-teljes Hamilton-kör problémára. □

1.2. A részgráf-izomorfia probléma adatbázisokban és az előszűrési módszerek

1.4. Definíció (Részgráf-izomorfia probléma adatbázisokban) *Adott egy* G_q *query gráf és* D*, egy címkézett gráfokat tartalmazó adatbázis. Határozzuk meg azon adatbázisbeli gráfokat, melyek tartalmazzák a query gráfot:* $\{G_t \,|\, G_t \in D \wedge G_q \subseteq G_t\}$.

1.2. Példa

Tekintsük az 1.2 ábrán látható G_q gráfot, D pedig álljon az 1.3 ábrán látható gráfokból. Ebben az esetben $G_q \subseteq G_1$ és $G_q \subseteq G_3$, de $G_q \not\subseteq G_2$.



1.3. ábra: A részgráf-izomorfia probléma adatbázisokban

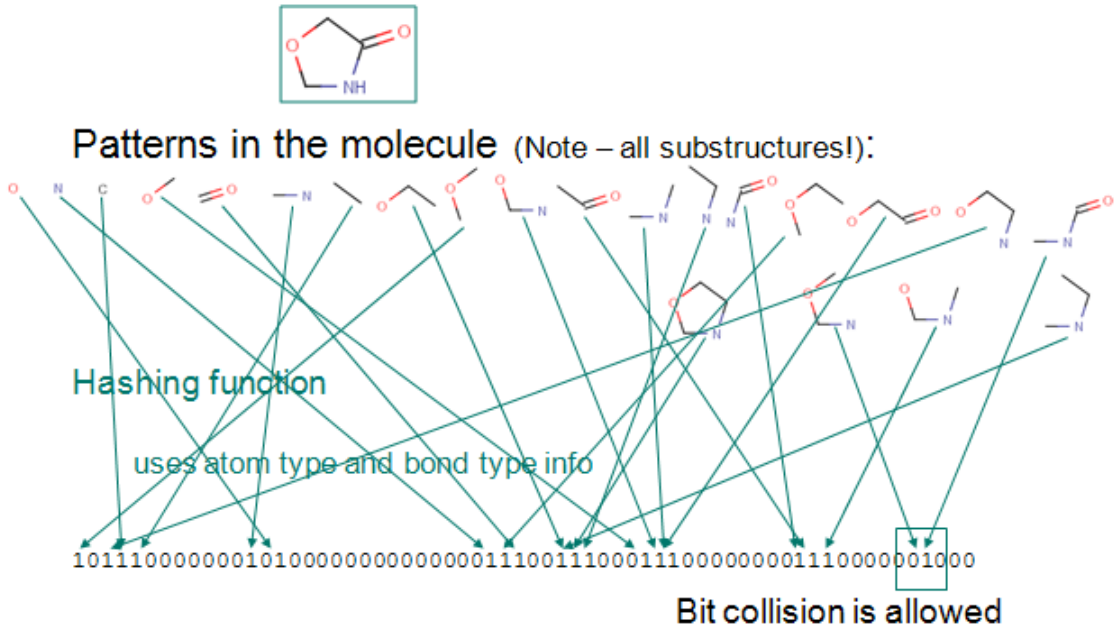
Figyelembe véve, hogy a gyakorlatban az adatbázis általában több millió struktúrát tartalmaz és a részgráf-izomorfia feladat NP-teljes probléma, ezért gyakran alkalmazott módszer, hogy az adatbázison előszűrést végeznek. Ezen előszűrési lépés célja minél több olyan adatbázisbeli gráf meghatározása, melyek biztosan nem tartalmazhatják részstruktúraként a query molekulát. Ezen lépés során a molekulákat fingerprintekkel írják le, melyek az egyes molekulákat jól jellemzik. A részgráf-izomorfia vizsgálatot az ezen előszűrési eljárást követően megmaradt target gráfokra végzik el, ezáltal csökkentve az időigényes műveletek számát. Jelen dolgozatban nem célunk az egyes előszűrési módszerek részletes ismertetése; csupán néhány eljárást mutatunk be a teljesség igénye nélkül, melyekre az egyes, részgráf-izomorfiát megoldó algoritmusok tárgyalásánál és összehasonlításánál hivatkozunk. Az olvasó bővebb információt találhat az előszűrési módszerekről [13]-ban, [16]-ban, [17]-ben, [24]-ben, [32]-ben, [37]-ben és [38]-ban.

Kémiai fingerprint

A kémiai informatika területén gyakori eljárás, hogy minden adatbázisbeli molekulagráfhoz kiszámítunk egy fingerprintet az előfeldolgozási fázisban (pl. a molekulák adatbázisba történő beszúrása során), és ezen értékeket a molekulákkal együtt az adatbázisban tároljuk. A kémiai fingerprint[2] [16] jellemzően egy 512-2048 hosszú, 0-kból és 1-esekből álló sorozat, melynek i. bitje egy tulajdonság jelenlétét vagy

[2] Chemical fingerprint

hiányát jelöli. Gyakran alkalmazott eljárás, hogy a molekulák azon részstruktúráit határozzuk meg, melyek atomszáma kisebb egy előre definiált korlátnál (értéke általában 5-7), majd ezen részstruktúrákra egy hash függvényt alkalmazva meghatározzuk azon bitpozíciókat, amelyeknek megfelelő helyeken a fingerprint értékét 1-re módosítjuk. Mivel a hash függvény alkalmazása során előfordulhatnak ütközések, ezért minden részstruktúrához érdemes 2-3 bitpozíciót is meghatározni, ahol a fingerprint értékének változtatása történik. Ezen fingerprintszámítási folyamatot tekinthetjük meg egy egyszerű molekula esetén az 1.4. ábrán [16].



1.4. ábra: Kémiai fingerprint (Forrás: [16])

A fenti módszer egy módosított változata esetén nem az adott molekula kis méretű részstruktúráinak, hanem előre meghatározott kisméretű molekulák jelenlétét vizsgáljuk, ahol ezen kisméretű molekulák függetlenek az adott molekulától. A kisméretű molekulák kiválasztásával foglalkozik [13], [37] és [38].

Egy „jó” kémiai fingerprintre teljesül, hogy ha a query molekula fingerprintjének i. pozícióján 1 szerepel, akkor az őt tartalmazó target molekula fingerprintjének i. bitje is 1. Ezen tulajdonságot felhasználva az alábbi módon végezhetünk előszűrést az adatbázison: az első lépésben határozzuk meg a query molekulához tartozó fingerprintet, majd ezen fingerprintet felhasználva, egyszerű bitműveleteket alkalmazva hasonlítsuk össze a target molekulák fingerprintjeivel, és szűrjük ki azon target molekulákat, melyek biztosan nem tartalmazzák részstruktúraként az adott query molekulát.

ECFP

Az ECFP[3]-t [17] a kémiai fingerprinttel ellentétben nem alkalmazható a részgráfizomorfia probléma esetén előszűrés céljából.[4] A fingerprint készítése iteratív eljárás, a kezdeti lépésében a molekula minden egyes atomjához egy, csupán az atom lokális környezetétől (pl. fokszám) függő címkét rendel hozzá. A későbbi lépésekben az atomok címkéit az alábbi módon változtatja meg: tekinti az adott atom szomszédainak az előző iteráció során meghatározott címkéit, majd ezen értékekből egy alkalmas hash függvény alkalmazásával egyetlen értékét számít ki, mely az atom új címkéjének felel meg. Ezáltal az egyes lépések során az algoritmus az atomok egyre nagyobb sugarú környezetét veszi figyelembe a címkék meghatározása során, mely az 1.5 ábrán is látható. Az algoritmus paraméterei közé tartozik k, mely az iterációk számát adja meg. Az algoritmus az iterációk során kapott atomcímkéket egyetlen halmazba gyűjti össze, melynek elemeit végül egy újabb hash függvény alkalmazásával egy 0-1 sorozatra képezi le, az így kapott sorozatot a molekula fingerprintjének nevezünk.

1.5. ábra: Az egyes iterációk alkalmával figyelembe vett atomok (Forrás: [17])

A dolgozat hátralévő részében az alábbi jelölési konvenciókat alkalmazzuk: u-val és u'-vel query csúcsot, v-vel és v'-vel pedig target csúcsot jelölünk. Ezenkívül a címkézőfüggvények esetén eltekintünk V_q, E_q illetve V_t, E_t alsó indexben történő feltüntetésétől feltüntetésétől, csupán a q illetve t jelölést alkalmazzuk; mivel a csúcs- és élcímkéző függvények a paraméterek számában eltérnek, így az adott környezetben egyértelműben kiderül, hogy a csúcs- vagy élcímkére hivatkozunk. A továbbiakban az élcímkék esetén a kettő zárójelpár helyett csupán egyetlen zárójelpárt tüntetünk fel, pl. $l_q((u,v))$ helyett $l_q(u,v)$-t írunk. Jelölje $deg(u)$ az u csúcs fokszámát, $adj(u)$ pedig az u szomszédait tartalmazó halmazt jelöli[5], továbbá $n_q := |V_q|, e_q := |E_q|$ és $n_t := |V_t|, e_t := |E_t|, n := n_q + n_t, \Delta_q(u) := \max_{u \in V_q} deg(u)$ és $\Delta_t(v) := \max_{v \in V_t} deg(v)$.

[3] Extended Connectivity Fingerprint

[4] Az ECFP gyakran alkalmazott fingerprint a gráfizomorfia probléma és molekulák hasonlóságának meghatározása esetén.

[5] $deg(u)$ és $adj(u)$ esetén eltekinthetünk a gráf alsó indexben történő jelölésétől, mivel feltettük, hogy $V_q \cap V_t = \emptyset$.

2. fejezet

Visszalépéses keresésen alapuló módszerek

A fejezetben röviden áttekintjük a részgráf-izomorfia probléma megoldására alkalmazott, visszalépéses keresésen alapuló módszereket. Ezen algoritmusokat a korábbi tanulmányok során már részletesen vizsgáltunk [32, 30, 31]. Célunk, hogy a részgráf-izomorfia problémát egy más megközelítésből, CSP feladatként modellezve oldjuk meg és hasonlítsuk össze a korábbi módszerekkel. Jelen fejezetben nem célunk a korábban elemzett módszerek részletes tárgyalása, csupán a későbbi összehasonlításhoz szükséges mélységben ismertetjük ezen algoritmusokat.

Az Ullmann-algoritmus (1976) [29] és a VF2 (2001) [8] visszalépéses keresésen alapuló módszerek, melyek futásuk során egy-egy keresési fát dinamikusan – a teljes fa exponenciális méretű lehet, csak a bejárás során szükséges részeket – építenek fel. A keresési fa egy csúcsa egy $f : V_q \to V_t$ parciális izomorfizmusnak felel meg. Ezen módszerek célja, hogy az aktuális leképezést bővítve részgráf-izomorfizmust határozzanak meg; vagy a keresési fa lehető legnagyobb olyan részét töröljék, melyben az aktuális parciális izomorfizmust bővítve ilyen függvény biztosan nem létezik. Az algoritmusok a keresési fa egyes részeinek törléséhez ún. „Finomítási kritériumokat" alkalmaznak.

A QuickSI (quick subgraph isomorphism, 2008) [24] szintén egy visszalépéses keresésen alapuló módszer – mely a másik két módszertől eltérően – a kezdő lépésben a query molekula várhatóan optimális atomsorrendjét határozza meg, majd az így kialakult sorrend alapján építi fel a keresési fát. A módszer célja olyan sorrend kialakítása, melynek alkalmazása során a keresési fa várhatóan minél kisebb. Az így kialakított sorrendre teljesül a VF2 során elvárt tulajdonság is, ezért az Ullmann- és a VF2 algoritmusokat ezen új sorrendre alkalmazhatjuk. Ezen atomsorrend nem a target molekuláktól, hanem az adatbázisbeli atom- és kötéstípusok gyakoriságától

függ; ennek köszönhetően elegendő a queryhez tartozó várhatóan optimális atomsorrendet egyetlen egyszer, a kezdeti lépés során meghatározni. Az így kialakított atomsorrendet végül az összes, páronkénti részstruktúra-vizsgálat során alkalmazhatjuk.

2.1. Az Ullmann algoritmus

Az algoritmus kezdőlépésében minden $u_i \in V_q$ csúcshoz meghatározza a lehetséges képeit tartalmazó halmazt. Természetesen adódó feltétel, hogy $f(u_i) = v_j$ esetén $l_q(u_i) = l_t(v_j)$ és $\deg(u_i) \leqslant \deg(v_j)$ teljesül, így inicializáljuk $\mathcal{H}_i$-t a következőképpen:

$$\mathcal{H}_i := \{v_j \,|\, v_j \in V_t \wedge l_q(u_i) = l_t(v_j) \wedge \deg(u_i) \leqslant \deg(v_j)\}. \tag{2.1}$$

2.1. Példa

A 1.2. ábrán látható query és target molekula esetén az alábbi halmazokat kapjuk:

$$u_1 \mapsto \mathcal{H}_1 = \{v_2, v_4, v_6\}$$
$$u_2 \mapsto \mathcal{H}_2 = \{v_2, v_4\}$$
$$u_3 \mapsto \mathcal{H}_3 = \{v_3, v_5\}$$
$$u_4 \mapsto \mathcal{H}_4 = \{v_2, v_4, v_6\}$$

Legyen $u_1, u_2, \ldots, u_{n_q}$ a G_q csúcsainak egy tetszőleges sorrendje. A keresési fa 0. szintje legyen az üres halmaz. A fa minden egyes csúcsa egy $f : V_q{}' \to V_t{}'$ injektív leképezés, ahol $V_q{}' \subseteq V_q$ és $V_t{}' \subseteq V_t$. Egy tetszőleges, i. szinten lévő leképezés az alábbi módon írható le:

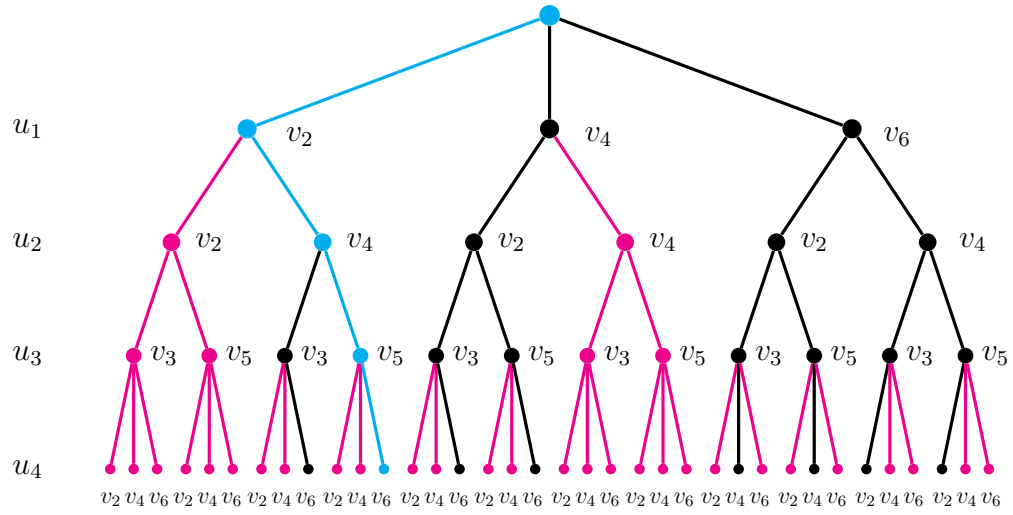
$$\begin{aligned} f(u_1) &= v_{j_1} \in \mathcal{H}_1 \\ &\vdots \\ f(u_i) &= v_{j_i} \in \mathcal{H}_i \end{aligned}$$

Ezen leképezést próbáljuk meg kiterjeszteni az $f(u_{i+1}) := v \in \mathcal{H}_{i+1}$ megfeleltetéssel úgy, hogy a következő tulajdonságok teljesüljenek, azaz az Ullmann-algoritmus az alábbi finomítási kritériumokat alkalmazza:

1. f injektív marad: $\forall k \in \{1, \ldots, i\}\,(v \neq v_{j_k})$
2. léteznek élek a megfelelő képek között és az élcímkék is megegyeznek:

$$\forall k \in \{1, \ldots, i\}\,((u_k, u_{i+1}) \in E_q \Rightarrow ((f(u_k), f(u_{i+1})) \in E_t \wedge l_q(u_k, u_{i+1}) = l_t(v_{j_k}, v)))$$

3. u_{i+1} minden szomszédjának lehet olyan képe, amely szomszédja v-nek.



2.1. ábra: A keresési fa az 1.2. ábrán látható G_q és G_t gráfok esetén

2.2. Példa

Tegyük fel, hogy az algoritmust az 1.2. ábrán G_q és G_t gráfokra alkalmazzuk. Az ezen inputhoz és az u_1, u_2, u_3, u_4 atomsorrendhez tartozó keresési fát a 2.1. ábrán láthatjuk. Az injektivitási feltétel miatt a fából törölhetjük bíborvörös színnel jelölt csúcsokat. A fában kék szín jelöli azon parciális izomorfizmusokat, melyeket bővítve részgráf-izomorfizmust kapunk.

Az algoritmus alapvető lépése az M mátrix ritkítása, mely az egyes $\mathcal{H}_i$ halmazokat tárolja. Abban az esetben, ha $v_j \in \mathcal{H}_i$, akkor az algoritmus megvizsgálja, hogy az u_i csúcs szomszédainak lehetséges képei között található-e olyan csúcs, mely szomszédja v_j-nek. Amennyiben ezen feltétel nem teljesül, töröljük v_j-t $\mathcal{H}_i$-ből és ismételten megpróbáljuk a mátrixot ritkítani.

Az Ullmann-algoritmus memóriaigénye eredetileg $O(n^3)$, mely egy apró észrevétel alapján $O(n^2)$-re csökkenthető [31].

2.2. A VF2 algoritmus

A VF2 algoritmus az aktuális parciális izomorfizmus kibővítéséhez először meghatározza *Jelöltek*(s)-t, mely azon (u, v) párokat tartalmazza, melyekkel ezen s állapotot bővíthetné. A *Jelöltek*(s)-beli párokra nem teljesül, hogy az s állapotot tetszőleges párral bővítve továbbra is (parciális) izomorfizmust kapunk. Ezen feltétel ellenőrzéséhez az algoritmus finomítási kritériumokat alkalmaz.

Az algoritmus ismertetése során az alábbi jelöléseket alkalmazzuk. Legyen $M(s) := \{(u, v) \,|\, u \in V_q \wedge v \in V_t \wedge f(u) = v\}$ az aktuális parciális izomorfizmus-

algorithm 1 $UllmannAlgoritmus(G_q, G_t, d)$

Input : G_q: query gráf, G_t: target gráf, d: keresési mélység, értéke kezdetben 0

Output: találatok száma $\in \mathbb{N}$: az 1.2. definíciónak eleget tevő $f : V_q \to V_t$ függvények száma, értéke kezdetben 0

```
1: if d ≥ n_q then
2:    találatok száma := találatok száma + 1
3: end if
4: for each 1 ≤ j ≤ n_t do
5:    if FinomításiKritérium(u_d, v_j) then
6:       Adatok mentése és az M mátrix ritkítása
7:       UllmannAlgoritmus(G_q, G_t, d + 1)
8:       Adatok visszaállítása
9:    end if
10: end for each
```

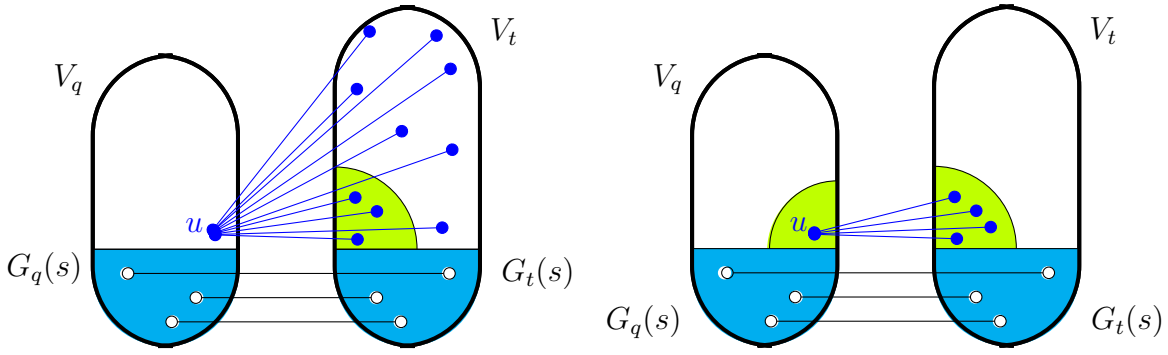
nak megfelelő párokat tartalmazó halmaz. Az $M(s)$-beli query és target csúcsokat $G_q(s)$-sel illetve $G_t(s)$-sel jelöljük. Továbbá jelölje $N(G_q(s))$ a $G_q(s)$-beli csúcsok szomszédainak halmazát, azaz $N(G_q(s)) := \bigcup_{u \in G_q(s)} adj(u)$. $N(G_t(s))$ hasonlóan definiálható.

A következő néhány ábrán a $G_q(s)$ és a $G_t(s)$ halmazokat kék, az $N(G_q(s))\backslash G_q(s)$ és az $N(G_t(s))\backslash G_t(s)$ halmazokat pedig zöld színnel szemléltetjük.

A $Jelöltek(s)$ meghatározásakor kétféle esetet különböztetünk meg, melyeket a 2.2 ábrán is illusztrálunk.

1. eset: $N(G_q(s))\backslash G_q(s) = \emptyset$
   $V_q\backslash G_q(s)$ legkisebb sorszámú csúcsának keressük a képét. Ezen csúcs szomszédainak képe nem ismert, emiatt a lehetséges képek halmazának szűkítésére nincs lehetőség, így ezen esetben $Jelöltek(s) := \{\min(V_q\backslash G_q(s))\} \times (V_t\backslash G_t(s))$.

2. eset: $N(G_q(s))\backslash G_q(s) \neq \emptyset$
   $N(G_q(s))\backslash G_q(s)$ legkisebb sorszámú csúcsának képét keressük, mely az élekre vonatkozó feltételek miatt csak $N(G_t)\backslash G_t(s)$-beli csúcs lehet, így $Jelöltek(s) := \{\min(N(G_q(s))\backslash G_q(s))\} \times (N(G_t(s))\backslash G_t(s))$.

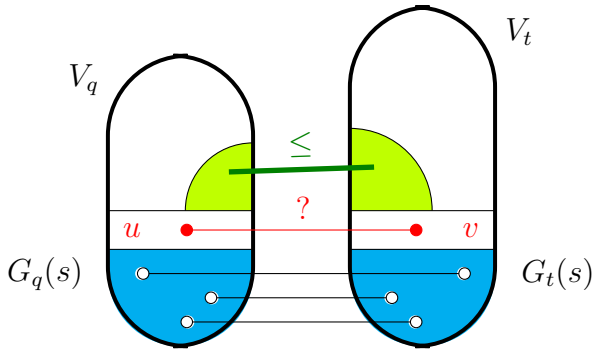


(a) $N(G_q(s))\backslash G_q(s) = \emptyset$ (b) $N(G_q(s))\backslash G_q(s) \neq \emptyset$

2.2. ábra: A $Jelöltek(s)$ meghatározásának két esete

Az algoritmus az alábbi finomítási kritériumot alkalmazza:

1. $l_q(u) = l_t(v)$
2. $\forall(u, u') \in E_q\,(u' \in G_q(s) \Rightarrow l_q(u, u') = l_t(v, f(u')))$
3. $|N(G_q(s) \cup \{u\})\backslash(G_q(s) \cup \{u\})| \leqslant |N(G_t(s) \cup \{v\})\backslash(G_t(s) \cup \{v\})|$, azaz $G_t(s')$-nek legalább annyi még le nem illesztett szomszédja kell, hogy legyen, mint $G_q(s')$-nek, melyet a 2.3. ábrán vizuálisan is szemléltetünk.



2.3. ábra: A finomítási kritérium 3. feltétele

A VF2 algoritmus memóriaigénye $O(n)$ [31].

2.3. A QuickSI algoritmus

A QuickSI algoritmus alapvető ötlete, hogy a query gráf atomsorrendjét módosítsuk úgy, hogy a visszalépéses keresés során felépített fa a lehető legkisebb legyen. Tekintettel arra, hogy molekulagráfok esetén az egyes csúcs- és élcímkék eloszlása

algorithm 2 $VF2Algoritmus(s)$

Input : G_q: query gráf, G_t: target gráf, d: keresési mélység, értéke kezdetben 0

Output: találatok száma $\in \mathbb{N}$, azaz az 1.2. definíciónak eleget tevő $f : V_q \to V_t$ függvények száma, értéke kezdetben 0

```
1: if d ≥ n_q then
2:    találatok száma := találatok száma + 1
3: else
4:    Jelöltek(s) meghatározása
5:    for each (u, v) ∈ Jelöltek(s) do
6:       if FinomításiKritérium(s, u, v) igaz then
7:          M(s') := M(s) ∪ {(u, v)}
8:          Adatok mentése
9:          VF2Algoritmus(s')
10:         Adatok visszaállítása
11:      end if
12:   end for each
13: end if
```

nem egyenletes, ezért célszerű először a ritkábban előforduló címkéjű csúcsok (pl. „Cl”, „F”) képét meghatározni, majd a gyakoriak (pl. „C”, „O”) képével foglalkozni.

Az eredeti algoritmus – némi módosítást követően – több komponensből álló, izolált csúcsokat tartalmazó gráf esetén is alkalmazható [31]. A módszer kezdőlépésében meghatározza az egyes címkék gyakoriságát a kereséshez használt adatbázisban, majd ezen értékek alapján egy súlyozott gráfot állít elő a query gráfból. Az így kapott gráf egyes komponensein a Prim-algoritmus [9] egy módosított változata[1] alapján határozza meg a query csúcsok feldolgozási sorrendjét. Ezt követően az Ullmann-algoritmust és a VF2-t ezen új atomsorrend szerint futtatjuk. A QuickSI algoritmus részletes leírása megtalálható [24]-ben és [31]-ben.

[1] Abban az esetben, ha a Prim-algoritmus tetszőlegesen választhatna a minimális súlyú élek közül, a QuickSI több heurisztikát is alkalmaz a remélhetően optimális él kiválasztása érdekében.

3. fejezet

A részgráf-izomorfia probléma CSP feladatként történő megoldása

3.1. A kényszerkielégítési probléma

3.1. Definíció (CSP feladat) *A CSP feladat* [1] *alatt egy* $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ *hármast értünk, ahol*

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ *a változókat tartalmazó halmaz*
- $\mathcal{D} = \langle D(x_1), \ldots, D(x_n) \rangle$ *az egyes változók lehetséges értékeit tartalmazó halmaz*
- $\mathcal{C}$*: kényszereket tartalmazó halmaz, ahol minden elem a változók egy részhalmazára vonatkozó kényszert ír le.*

A CSP feladat megoldása alatt a változók egy olyan értékadását értjük, mely valamennyi kényszert kielégít.

CSP feladatként számos ismert probléma leírható, pl. a gráfszínezés, a gráfizomorfia, a Sodoku és az n-királynő probléma.

3.2. Tétel *Annak eldöntése, hogy egy adott CSP feladatnak létezik-e megoldása, NP-teljes feladat.*

3.2. A részgráf-izomorfia probléma CSP feladatként

A részgráf-izomorfia problémát az alábbi módon fogalmazhatjuk meg CSP feladatként, mely egyben azt is bizonyítja, hogy a CSP NP-teljes feladat. A query

[1] Constraint satisfaction problem, melyet magyarra „kényszerkielégítési probléma”-ként fordíthatnánk. A továbbiakban CSP-ként hivatkozunk rá.

gráf minden egyes u csúcsának rendeljünk hozzá egy x_u változót. Jelölje $x_u := v$ ($v \in V_t$) azt, hogy $f(u) = v$, azaz az u query csúcsot a v target csúcsra képeztük le. Ekkor az 1.2. definícióban (5. oldal) szereplő feltételeket az alábbi módon írhatjuk le:

1. a leképező függvény injektív: $\forall u, u' \in V_q \, (u \neq u' \Rightarrow x_u \neq x_{u'})$
2. a leképezés megőrzi a csúcscímkéket: $\forall u \in V_q \, (l_q(u) = l_t(x_u))$
3. a leképezés éltartó: $\forall u, u' \in V_q \; ((u, u') \in E_q \Rightarrow (x_u, x_{u'}) \in E_t)$
4. a leképezés élcímketartó: $\forall u, u' \in V_q \; ((u, u') \in E_q \Rightarrow l_q(u, v) = l_t(x_u, x_{u'}))$.

A $D(x_u)$ halmaz tulajdonképpen az u query csúcs lehetséges képeit tartalmazó halmaz, mely melyet az Ullmann-algoritmusnál $\mathcal{H}_i$-vel ((2.1), 11. oldal) jelöltünk. Mivel a keresési tér exponenciálisan nagy lehet, ezért célunk, hogy az inicializálás során a $D(x_u)$ halmazok minél gyorsabban meghatározhatóak legyenek, ugyanakkor lehetőleg minél kevesebb elemet tartalmazzanak. Gyakran alkalmazott módszer, hogy ezen halmazok inicializálása során csupán a fokszámot és a csúcscímkét veszik figyelembe, így $D(x_i) := \{v_j \,|\, v_j \in V_t \wedge l_q(u_i) = l_t(x_j) \wedge \deg(u_i) \leq \deg(v_j)\}$.

Habár úgy tűnhet, hogy a részgráf-izomorfia probléma CSP feladatként történő megfogalmazását követően a továbbiakban nincs szükség a query és a target molekulagráfokra, számos, speciálisan a részgráf-izomorfiát megoldó módszer (pl. LV2002, LAD, ILF(k)) esetén létfontosságú ezen molekulagráfok szerkezetének pontos ismerete (jellemzően egy adott csúcs szomszédainak meghatározása a feladat).

3.3. Definíció (AllDiff) *$AllDiff(S)$ jelentse azon kényszert, hogy S-beli elemek páronként különböző értéket vesznek fel.*

3.1. Példa

Tekintsük az 1.2. ábrán (6. oldal) látható G_q és G_t gráfot. Ezen esetben a részgráf-izomorfia problémát következőképpen írhatjuk le CSP feladatként:

- $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$
- $\mathcal{D} = \langle D(x_1), D(x_2), D(x_3), D(x_4)\rangle$, *ahol* $D(x_1) = \{v_1, v_2, v_4, v_5, v_6\}$, $D(x_2) = \{v_2\}$, $D(x_3) = \{v_3\}$ *és* $D(x_4) = \{v_1, v_2, v_4, v_5, v_6\}$
- *a $\mathcal{C}$ kényszerhalmaz a következő elemekből áll:*
  - $l_t(x_1) = l_t(x_2) = l_t(x_4) =$ „C”
  - $l_t(x_3) =$ „N”

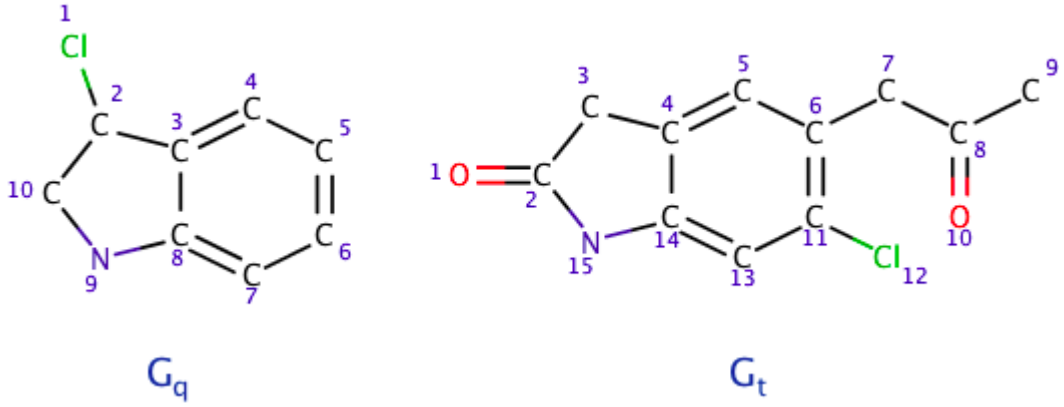
- $(x_1, x_2) \in E_t \wedge l_t(x_1, x_2) =$ *„kettős kötés”*
- $(x_2, x_3) \in E_t \wedge l_t(x_2, x_3) =$ *„egyszeres kötés”*
- $(x_2, x_4) \in E_t \wedge l_t(x_2, x_4) =$ *„egyszeres kötés”*
- $AllDiff(\{x_1, x_2, x_3, x_4\})$

Ezen CSP feladat egy lehetséges megoldása:

$$x_1 = 2 \qquad x_3 = 5$$
$$x_2 = 4 \qquad x_4 = 6$$

3.2. Példa

A továbbiakban, az egyes módszereket a 3.1 ábrán látható query és target gráfon alkalmazva mutatjuk be. A fentiek alapján könnyen megfogalmazhatjuk ezen részgráf-izomorfia feladatot CSP feladatként, melytől jelen esetben eltekintünk, csupán a kezdeti $D(x_u)$ halmazokat ismertetjük.



3.1. ábra: A módszerek ismertetése során alkalmazott G_q és G_t példagráfok

$D(x_1) = \{v_9, v_{12}\}$
$D(x_2) = \{v_2, v_4, v_6, v_8, v_{11}, v_{14}\}$
$D(x_3) = \{v_2, v_4, v_6, v_8, v_{11}, v_{14}\}$
$D(x_4) = \{v_2, v_3, v_4, v_5, v_6, v_8, v_{11}, v_{13}, v_{14}\}$
$D(x_5) = \{v_2, v_3, v_4, v_5, v_6, v_8, v_{11}, v_{13}, v_{14}\}$
$D(x_6) = \{v_2, v_3, v_4, v_5, v_6, v_8, v_{11}, v_{13}, v_{14}\}$
$D(x_7) = \{v_2, v_3, v_4, v_5, v_6, v_8, v_{11}, v_{13}, v_{14}\}$
$D(x_8) = \{v_2, v_4, v_6, v_8, v_{11}, v_{14}\}$
$D(x_9) = \{v_7, v_{15}\}$
$D(x_{10}) = \{v_2, v_3, v_4, v_5, v_6, v_8, v_{11}, v_{13}, v_{14}\}$

A szakirodalomban számos CSP-t megoldó módszer található meg. Az általunk vizsgált, a részgráf-izomorfiát megoldó algoritmusok tulajdonképpen visszalépéses keresések, melyek az egyes értékadást követően különböző szűrési erejű feltételek

teljesülését ellenőriznek, hasonlóan az Ullmann-algoritmusnál és a VF2-nél tárgyalt „finomítási kritériumokhoz". Ezen módszerek közös célja, hogy az egyes $D(x_u)$ halmazokból minél több olyan elemet eltávolítsanak, melyek nem tartoznak egyetlen megoldáshoz sem. Tekintettel arra, hogy a dolgozatban címkézett gráfokkal foglalkozunk, ezért a csúcs- és élcímkék egyezőségének vizsgálatával a $D(x_u)$ halmazokat tovább szűkíthetjük. Az egyes módszerek leírásánál meghatározzuk azok időigényét, látni fogjuk, hogy ezek között nagyságrendi különbségek is előfordulhatnak.

Az egyes szűrési módszerek erősségének összehasonlítása során szükségünk lesz az alábbi definícióra.

3.4. Definíció *Legyen A és B két tetszőleges szűrési módszer. Az A módszert (szigorúan) erősebbnek nevezzük a B módszernél, ha az A módszer által meghatározott (u, v) $(u \in V_q \wedge v \in D(x_u))$ inkonzisztens párok halmaza (szigorúan) bővebb.*

Számos szűrési eljárás esetében előfordul, hogy a $D(x_u)$ halmazból való törlést követően ismételt ellenőrzés szükséges, hiszen korábban teljesülő feltételek romolhattak el a törlés következtében. Amennyiben ezen lépés indokolt az egyes eljárás során, külön jelezzük.

A fenti algoritmusok további közös tulajdonsága, hogy minden lépésben ellenőrzik, hogy valamely $D(x_u)$ halmaz üressé vált-e. Ebben az esetben visszalépnek az előző állapotra, hiszen az x_u változónak nem lehetséges értéket adni, mely konzisztens lenne az aktuális állapottal.

A továbbiakban először néhány általános eljárással foglalkozunk, majd a számos, speciálisan a részgráf-izomorfia probléma megoldására publikált technikákat tekintünk át.

3.3. Általános módszerek

3.3.1. Forward checking és arc-consistency

Az FC[2] alapötlete, hogy az $x_u := v$ értékadást követően valamennyi $x_{u'}$-re, melynek az értéke egyelőre nem ismert, ellenőrizzük, hogy a $x_{u'} := v'$ $(v' \in D(x_{u'}))$ esetén teljesülnek-e az x_u és $x_{u'}$ közötti kényszerek. Amennyiben valamelyik kényszer sérül, töröljük v'-t $D(x_{u'})$-ből.

A fentinél erősebb szűrési módszert kaphatunk az alábbi módon: tekintsünk egy tetszőleges x_u változót és egy $v \in D(x_u)$ értéket. Az x_u változó a v értékre és az $x_{u'}$ $(x_{u'} \neq x_u)$ változóra való támogatottsága alatt egy olyan $v' \in D(x_{u'})$ értéket értünk, hogy az $x_u := v \wedge x_{u'} := v'$ szimultán értékadás esetén teljesülnek a

[2] forward checking

két változóra előírt feltételek. Egy CSP-t AC-nek[3] nevezünk, ha tetszőleges x_u-t, $v \in D(x_u)$-t és $x_{u'}$-t kiválasztva létezik ilyen támogatottság. Amennyiben nem létezik a kívánt tulajdonságú támogatottság, töröljük v-t $D(x_u)$-ból, hiszen az $x_u := v$ értékadás esetén nem lehetséges $x_{u'}$ értékének meghatározása.

Lényeges különbség az FC és az AC között, hogy az FC egyszerre csak egy olyan x_u változót tekint, melynek értéke még nem ismert, ezzel szemben az AC módszer ilyen változókból álló párokra vonatkozó feltételt ellenőriz.

A query csúcsok képeire vonatkozó injektivitási kényszert[4] tekinthetünk FC és AC esetén is, az így kapott módszerek FC(Diff) és GAC(AllDiff). Az FC(Diff) eljárás az $x_u := v$ értékadást követően törli v-t azon $D(x_{u'})$ halmazokból, melyekre u' képe még ismert. Ezen módszer időigénye $O(n_q)$. A GAC(AllDiff)[5] eljárás megvizsgálja, hogy $x_u := v$ esetén lehetséges-e valamennyi query csúcshoz páronként különböző target csúcsot rendelni. A GAC-módszer általánosításának tekinthető a LAD algoritmus, mely az egyes leképezéseket nem globálisan, hanem lokális szinten tekinti és melyet a későbbiekben (3.4.3. fejezet, 28. oldal) részletesen tárgyalunk.

Az előzőekben csak a csúcsokra vonatkozó feltételekkel foglalkoztunk, az éleket nem vizsgáltunk. A fentieknél erősebb szűrési feltétel az FC(Edge) és az AC(Edge). Az FC(Edge) módszer eredetileg az $x_u := v$ értékadást követően, az élmegmaradási kényszer alapján az egyes lehetséges képek halmazát az alábbiak szerint módosítja:

$$D(x_{u'}) := D(x_{u'}) \cap adj(v) \quad \forall u' \in adj(u),$$

mely $O(\deg(u) \cdot n_t)$ idő alatt elvégezhető. Tekintettel arra, hogy az általunk vizsgált molekulagráfok címkézettek, ezért a fentieken túl v' abban az esetben is törölhető $D(x_{u'})$-ből, amennyiben $l_q(u, u') \neq l_t(v, v')$.

Megjegyezzük, hogy ezen módszer az Ullmann-algoritmusnál és a VF2-nél is alkalmazható „szülő heurisztika" [31] általánosításaként fogható fel. A „szülő heurisztika" esetén a query csúcsokhoz egy-egy query csúcsot, ún. „szülő csúcs"-ot tartunk nyilván, melyet $p(u)$-val jelölünk. Ezen „szülő csúcs"-ra teljesül, hogy $(u, p(u)) \in E_q$ és a $p(u)$ query csúcs képét az aktuálisan vizsgálandó csúcsot megelőzően már rögzítettünk. A keresési teret x_u meghatározásakor, a $p(u)$ „szülő csúcs" értéket figyelembe véve szűkíthetjük: ismert, hogy $(u, p(u)) \in E_q$ és $x_{p(u)}$ már rögzített, így x_u lehetséges képei csak $x_{p(u)}$ szomszédjai közül kerülhetnek ki. Ezen gondolatmenetet általánosítja az FC(Edge) módszer, hiszen egy tetszőleges u csúcs képének meghatározását megelőzően már számos olyan u' képét rögzíthettük, melyek szomszédosak u-val. Az egyes $x_{u'}$ értékek meghatározását követően az FC(Edge) módszer $D(x_u)$-t

[3] arc consistent

[4] difference constraint

[5] global all-different

is módosíthatta, mivel szükséges feltétel, hogy $x_u \in adj(u')$. Így amennyiben korábban u legalább kettő szomszédjának képét már fixáltuk, úgy $D(x_u)$ nem lehet bővebb annál a $D'(x_u)$ halmaznál, melyet „szülő heurisztikát" alkalmazva kapnánk.

A fentinél is erősebb szűrési módszert, az AC(Edges)-t kapunk, ha a $D(x_u)$ halmazokat az alábbi feltételt ellenőrizve próbáljuk meg szűkíteni:

$$\forall (u, u') \in E_q \; \forall v \in D(x_u) \; \exists v' \in D(x_{u'}) \; ((v, v') \in E_t).$$

3.3. Példa

Tekintsük a 3.1. ábrán látható G_q és G_t gráfokat, és tegyük fel, hogy $x_8 := v_{14}$. Ebben az esetben az FC(Diff) eljárás a $D(x_u)$ halmazokat az alábbiképpen módosítja: $D(x_8) := \{14\}$ és törli v_{14}-et minden $D(x_u)$-ből $(u \neq 8)$.

Az FC(Edges) módszer megállapítja, hogy $adj(u_8) = \{u_3, u_7, u_9\}$ és $adj(v_{14}) = \{u_4, u_{13,}, u_{15}\}$, majd törli a $D(x_3)$, a $D(x_7)$ és a $D(x_9)$ halmazokból azon elemeket, melyek nem elemei a $\{4,13,15\}$ halmaznak.

A fenti két eljárás következtében $D(x_3) = \{4\}$, továbbá a kényszerek között megtalálhatóak a $(x_3, x_4) \in E_t$ és $l_t((x_3, x_4)) =$ „kettős kötés" kényszerek is, így az AC(Edges) törölni tudja $D(x_4)$-ből v_3-at, mivel ezen érték nem rendelkezik támogatottsággal ($l_q(u_3, u_4) =$ „kettős kötés" és $l_t(v_4, v_3) =$ „egyszeres kötés").

3.4. Speciális, részgráf-izomorfiát megoldó módszerek

A következőkben néhány speciális, részgráf-izomorfiát megoldó CSP módszert ismertetünk, melyek némi módosítást követően alkalmazhatóak akár a feszített részgráfként történő tartalmazás eldöntésére, vagy gráfizomorfia vizsgálatára is.

3.4.1. LV2002

Az LV2002 módszer az $x_u := v$ megfeleltetést megelőzően megvizsgálja u szomszédainak lehetséges képeit. Könnyen belátható, hogy $x_u = v$ csak abban az esetben lehet konzisztens, amennyiben u valamennyi u' szomszédjának lehet olyan v' képe, mely szomszédja v-nek. Ezen módszer az u csúcs szomszédainak lehetséges leképezéseit egyszerre vizsgálja halmazok számosságára vonatkozó feltétel ellenőrzésével. Ennek érdekében definiálja az $\mathcal{F}(u, v)$ halmazt, mely azon target csúcsokat tartal-

mazza, melyek szomszédai v-nek és szerepelnek valamely $u' \in adj(u)$-hoz tartozó $D(x_u)$-ban.

$$\mathcal{F}(u,v) := \begin{cases} \emptyset & \text{ha } \exists u' \in adj(u) : D(x_{u'}) \cap adj(v) = \emptyset \\ \left(\bigcup_{u' \in adj(u)} D(x_{u'}) \right) \cap adj(v) & \text{különben} \end{cases}$$

Szintén könnyen meggondolható, hogy $x_u = v$ csak akkor tartozhat a CSP feladat egy megoldásához, ha $|\mathcal{F}(u,v)| \geq adj(u)$. A módszer az $\mathcal{F}(u,v)$ definiálása során figyelembe veszi azon esetet is, melynek során u valamely u' szomszédjának nincs olyan lehetséges képe, mely szomszédja lenne v-nek. Ebben az esetben $x_u := v$ nyilván nem lehetséges, így ezen esetben is törölhető v $D(x_u)$-ból.

A fenti módszer időigénye $O(n_q^2 \cdot n_t^2)$.

3.4. Példa

Tekintsük a 3.1. ábrán látható G_q és G_t gráfokat és vizsgáljuk $\mathcal{F}(u_3, v_{14})$-et:

$$\mathcal{F}(u_3, v_{14}) = (D(x_2) \cup D(x_4) \cup D(x_8)) \cap \{v_4, v_{13}, v_{15}\} = \{v_4, v_{13}\},$$

továbbá $|\{v_4, v_{13}\}| = 2 < \deg(u_3) = 3$, így $x_3 := v_{14}$ értékadás nem lehet konzisztens.

Az LV2002 módszer hasonlít a VF2 finomítási kritériumai között szereplő, a szomszédsági halmazok számosságára vonatkozó feltétel ellenőrzéséhez. Fontos eltérés a két eljárás között, hogy míg az LV2002 egyetlen query csúcs szomszédainak lehetséges képeinek halmazát hasonlítja össze a target csúcs szomszédainak halmazával, addig a VF2 algoritmus a korábban leképzett query csúcsok szomszédait veti össze a target hasonló tulajdonságú csúcsaival.

3.4.2. ILF(k)

Az ILF(k) szűrési módszer egy iteratív algoritmus, melynek alapötlete a következő. A módszer a kezdőlépésében természetesen adódó és könnyen számítható címkéket rendel az egyes csúcsokhoz, majd ezen címkék halmazán egy parciális rendezést definiálva töröl elemeket a $D(x_u)$ halmazokból. Az eljárás egy későbbi iterációban az előző iteráció során meghatározott csúcscímkéket és rendezést felhasználva újabb csúcscímkéket rendel az egyes csúcsokhoz a szomszédos csúcsok előző címkéit felhasználva, majd az új címkéken definiál egy rendezést. Ezen új információk birtokában az eljárás az egyes $D(x_u)$ halmazokból újabb értékek törlésére tesz kísérletet.

Ezen gondolatmenet – mely szerint a query és a target csúcsok egyre nagyobb sugarú környezetét vizsgáljuk – megtalálható az ECFP (fingerprint) készítése során is. Fontos különbség az ILF(k) és az ECFP módszer között, hogy az ECFP nem használható a részgráf-izomorfia probléma esetén, ezzel szemben az ILF(k) eljárás az újabb csúcscímkéket oly módon határozza meg, hogy az továbbra is részgráf-izomorfizmus konzisztens maradjon.

Az algoritmus nevében szereplő k paraméter az iterációk számára vonatkozó felső korlátot jelöli. Az algoritmus nem feltétlen hajt végre k iterációt, hiszen előfordulhat, hogy valamely, k-nál kisebb sorszámú iteráció során valamely $D(x_u)$ halmaz üres halmazzá válik (ezen esetben a módszer az inkonzisztens állapotot detektálja, majd leáll, hiszen a megfelelő x_u változónak nem lehet értéket adni), vagy az algoritmus fixpontot ér el, azaz az újabb címkék bevezetésével nem lehetséges a $D(x_u)$ halmazok méretét csökkenteni.

Az algoritmus formális tárgyalását megelőzően bevezetünk néhány definíciót.

3.5. Definíció *Címkézés alatt egy $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ hármast értünk, ahol*

- $\mathbb{L}$ *a csúcscímkéket tartalmazó halmaz*
- $\preccurlyeq \subseteq \mathbb{L} \times \mathbb{L}$ $\mathbb{L}$*-en definiált részbenrendezés*
- $\alpha : V_q \cup V_t \to \mathbb{L}$ *függvény, mely az egyes query és target csúcsokhoz címkét rendel.*

3.6. Definíció *Az $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ címkézés szerint az (u, v) $(u \in V_q, v \in D(x_u))$ párt konzisztensnek nevezzük, ha $\alpha(u) \preccurlyeq \alpha(v)$ teljesül.*

3.7. Definíció *Részgráf-izomorf-konzisztens címkézés alatt egy olyan $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ címkézést értünk, ahol tetszőleges $f : V_q \to V_t$ részgráf-izomorfizmusra és tetszőleges u query csúcsra $\alpha(u) \preccurlyeq \alpha(f(u))$ teljesül.*

A részgráf-izomorfia probléma vizsgálata során gyakran alkalmazott kezdeti címkézések:

- $\ell_{deg} = (\mathbb{N}, \leq, deg)$, ahol $deg(u)$ az u csúcs fokszáma
- $\ell_{tav_k} = (\mathbb{N}, \leq, tav_k)$, ahol $tav_k(u)$ az u-tól legfeljebb k távolságra lévő csúcsok száma
- $\ell_{klikk_k} = (\mathbb{N}, \leq, klikk_k), ahol\ klikk_k(u)$ az u csúcsot tartalmazó k méretű klikkek számát jelöli.

Az általunk vizsgált molekulagráfok ritkán tartalmaznak K_3 vagy K_4 teljes gráfot részstruktúraként, ezért a ℓ_{klikk_k} címkézés alkalmazása molekulagráfok esetén kevésbé gyakori módszer.

3.8. Definíció *Az S alaphalmazon értelmezett multihalmaz egy $m : S \to \mathbb{N}$ függvény, ahol $m(s)$ az s elem multiplicitását adja meg.*

A multihalmaz a halmaztól eltérően egy elemet több példányban is tartalmazhat.

Jelölés: *$S = \{\{s_0, \ldots, s_0, \ldots, s_l, \ldots, s_l\}\}$, ahol az s_i elem a multiplicitásának megfelelő számban kerül ismétlésre.*

3.9. Definíció *Tegyük fel, hogy adott két különböző multihalmaz, m és m' az S alaphalmazon és egy $\preccurlyeq \subseteq S \times S$ részbenrendezés. $m \preccurlyeq m'$ pontosan akkor, ha létezik egy olyan $t : m \to m'$ injektív függvény, hogy minden $m_i \in m$ esetén $m_i \preccurlyeq t(m_i)$ teljesül.*

3.5. Példa

Ha tekintjük a természetes számok halmazán a $\leq$ rendezést, akkor $\{\{2,3,3\}\} \preccurlyeq \{\{1,3,4,4\}\}$, *de* $\{\{2,2,3\}\} \not\preccurlyeq \{\{1,3,4\}\}$.

3.10. Definíció *Egy adott $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ címkézés szomszédokra történő kiterjesztése alatt az $\ell' = (\mathbb{N}, \preccurlyeq', \alpha')$ címkézést értjük, ahol*

- $\mathbb{L}' = \mathbb{L} \cdot (\mathbb{L} \to \mathbb{N})$
- $\alpha'(u) = \alpha(u) \cdot \{\{\alpha(u') \mid u' \in adj(u)\}\}$
- *$l_1 \cdot m_1 \preccurlyeq l_2 \cdot m_2$ – ahol l_1 és l_2 $\mathbb{L}$ feletti címke, m_1 és m_2 pedig $\mathbb{L}$-beli elemekből álló multihalmaz – akkor és csak akkor, ha $l_1 \preccurlyeq l_2$ és $m_1 \preccurlyeq m_2$ teljesül.*

3.6. Példa

Tekintsük a 3.1 ábrán látható G_q és G_t gráfokat, és legyen $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ az ℓ_{deg} csúcscímkékkel történő kiegészítése, ahol

- $\mathbb{L} = \{\langle l_1, l_2 \rangle | \; l_1$ *egy periódusos rendszerbeli elem vegyjele* $\wedge \; l_2 \in \mathbb{N}\}$
- $\alpha(v) = \langle l(u), deg(u) \rangle$ *(ahol $l(u)$ az u csúcscímkéjét jelöli)*[6]

[6] Az ILF(k) módszert a szerzők címkézetlen gráfokra ismertették [35], melyet úgy módosítottunk, hogy a molekulagráfok csúcscímkéit is vegye figyelembe, ezáltal várhatóan erősebb szűrési eljárást kapunk.

- $\langle l_1, l_2 \rangle \preccurlyeq \langle l'_1, l'_2 \rangle$ *pontosan akkor, ha* $l_1 = l'_1$ *és* $l_2 \leq l'_2$ *teljesül.*

Ebben az esetben a query és a target csúcsok ℓ *címkéi (a továbbiakban az egyes címkékre a zárójelben feltüntetett jelöléssel hivatkozunk):*

$$\begin{aligned}
\alpha(u_1) = \alpha(v_{12}) &= \langle \text{„}Cl\text{”}, 1 \rangle && (m_1) \\
\alpha(u_4) = \alpha(u_5) = \alpha(u_6) = \alpha(u_7) = \alpha(u_{10}) &= \\
= \alpha(v_3) = \alpha(v_5) = \alpha(v_7) = \alpha(v_{13}) &= \langle \text{„}C\text{”}, 2 \rangle && (m_2) \\
\alpha(u_2) = \alpha(u_3) = \alpha(u_8) &= \\
= \alpha(v_2) = \alpha(v_4) = \alpha(v_6) = \alpha(v_8) = \alpha(v_{11}) = \alpha(v_{14}) &= \langle \text{„}C\text{”}, 3 \rangle && (m_3) \\
\alpha(u_9) = \alpha(v_{15}) &= \langle \text{„}N\text{”}, 2 \rangle && (m_4) \\
\alpha(v_1) = \alpha(v_{10}) &= \langle \text{„}O\text{”}, 1 \rangle && (m_5) \\
\alpha(v_9) &= \langle \text{„}C\text{”}, 1 \rangle && (m_6)
\end{aligned}$$

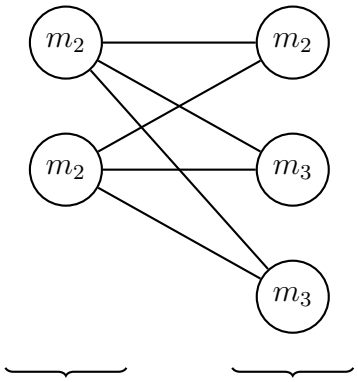
A címkék összehasonlítása pedig: $m_i \preccurlyeq m_i \quad (\forall i \in \{1,2,3,4\})$, $m_2 \preccurlyeq m_3$ *és* $m_6 \preccurlyeq \{m_2, m_3\}$. *A fenti címkézés szomszédokra történő kiterjesztése,* ℓ' *:*

$$\begin{aligned}
\alpha'(u_1) = m_1 \cdot \{\{m_3\}\} &\quad (n_1) \\
\alpha'(u_2) = \alpha'(v_{11}) = m_3 \cdot \{\{m_1, m_2, m_3\}\} &\quad (n_2) \\
\alpha'(u_3) = m_3 \cdot \{\{m_2, m_3, m_3\}\} &\quad (n_3) \\
\alpha'(u_4) = \alpha'(u_7) = m_2 \cdot \{\{m_2, m_3\}\} &\quad (n_4) \preccurlyeq' \{n_6, n_{11}, n_{12}, n_{13}\} \\
\alpha'(u_5) = \alpha'(u_6) = m_2 \cdot \{\{m_2, m_2\}\} &\quad (n_5) \preccurlyeq' \{n_3, n_4, n_6, n_{11}, n_{12}, n_{13}\} \\
\alpha'(u_8) = \alpha'(v_{14}) = m_3 \cdot \{\{m_2, m_3, m_4\}\} &\quad (n_6) \\
\alpha'(u_9) = m_4 \cdot \{\{m_2, m_3\}\} &\quad (n_7) \preccurlyeq' \{n_{14}\} \\
\alpha'(u_{10}) = m_2 \cdot \{\{m_3, m_4\}\} &\quad (n_8) \\
\alpha'(v_1) = \alpha'(v_{10}) = m_5 \cdot \{\{m_3\}\} &\quad (n_9) \\
\alpha'(v_2) = m_3 \cdot \{\{m_2, m_4, m_5\}\} &\quad (n_{10}) \\
\alpha'(v_3) = \alpha'(v_5) = \alpha'(v_{13}) = m_2 \cdot \{\{m_3, m_3\}\} &\quad (n_{11}) \\
\alpha'(v_4) = \alpha'(v_6) = m_3 \cdot \{\{m_2, m_2, m_3\}\} &\quad (n_{12}) \preccurlyeq' \{n_3\} \\
\alpha'(v_7) = m_2 \cdot \{\{m_3, m_3\}\} &\quad (n_{13}) \\
\alpha'(v_{15}) = m_4 \cdot \{\{m_3, m_3\}\} &\quad (n_{14}) \\
\alpha'(v_8) = m_3 \cdot \{\{m_2, m_5, m_6\}\} &\quad (n_{15}) \\
\alpha'(v_9) = m_6 \cdot \{\{m_3\}\} &\quad (n_{16}) \preccurlyeq' \{n_2, n_3, n_4, n_6, n_8, n_{11}, n_{12}, n_{13}, n_{14}\} \\
\alpha'(v_{12}) = m_1 \cdot \{\{m_3\}\} &\quad (n_{17})
\end{aligned}$$

A címkézés kiterjesztése során, az összehasonlítható címkék meghatározása érdekében az ILF(k) módszer[7] *egy páros gráfot készít el min-*

[7] Hasonló gondolatmenet található meg a LAD algoritmus esetén is, melyet a későbbiekben ismertetünk.

den címkepárra. Tegyük fel, hogy az eljárás el szeretné dönteni, hogy $n_5 = m_2 \cdot \{\{m_2, m_2\}\} \preccurlyeq n_3 = m_3 \cdot \{\{m_2, m_3, m_3\}\}$ *teljesül-e. Az előző iteráció eredményéből már rendelkezésre áll, hogy* $m_2 \preccurlyeq m_3$*; csupán csak azt szükséges vizsgálni, hogy* $\{\{m_2, m_2\}\} \overset{?}{\preccurlyeq} \{\{m_2, m_3, m_3\}\}$*. Ennek érdekében az algoritmus egy* $G = (A = \{\{m_2, m_2\}\}, B = \{\{m_2, m_3, m_3\}\}, E)$ *páros gráfot épít fel, ahol* $(i, j) \in E$ $(i \in A, j \in B) \Leftrightarrow i \preccurlyeq j$*. Az így definiált* G *gráf tekinthető meg a 3.2 ábrán.* $m \preccurlyeq m'$ *pontosan akkor teljesül, ha a* G *gráfban létezik* m*-t fedő párosítás. Ezen kérdés megválaszolható a Hopcroft-Karp algoritmus [9] alkalmazásával, melynek futásideje* $O(|E| \cdot \sqrt{|V|}) = O\left(\Delta_q \cdot \Delta_t \cdot \sqrt{\Delta_q + \Delta_t}\right) = O(\Delta_q \cdot \Delta_t^{3/2})$*. Ezen páros gráfok száma megegyezik az* (u, v) $(u \in V_q \wedge v \in D(x_u))$ *párok számával, mely felülről becsülhető* $O(n_q \cdot n_t)$*-vel.*



$A = \{\{m_2, m_3, m_3\}\}$ $B = \{\{m_2, m_3, m_3\}\}$

3.2. ábra: Multihalmazok összehasonlítása

3.11. Tétel *Legyen* $\ell' = (\mathbb{L}', \preccurlyeq', \alpha')$ *a* $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ *címkézés szomszédokra történő kiterjesztése. Amennyiben* ℓ *részgráf-izomorf-konzisztens, akkor* ℓ' *is részgráf-izomorf-konzisztens és* ℓ' *legalább olyan erős szűrési módszer, mint* ℓ*.*

Bizonyítás. Tekintsünk egy tetszőleges f részgráf-izomorfizmust, egy ℓ részgráf-izomorf-konzisztens címkézést és egy u query csúcsot. Meg kell mutatnunk, hogy $\alpha(u) \cdot \{\{\alpha(u') \mid u' \in adj(u)\}\} \preccurlyeq' \alpha(f(u)) \cdot \{\{\alpha(v') \mid v' \in adj(f(u))\}\}$. Mivel f részgráf-izomorf-konzisztens címkézés, ezért $\alpha(u) \preccurlyeq \alpha(f(u))$; továbbá az f leképezés eleget tesz a 3.9. definícióban szereplő feltételeknek az $m = \{\{\alpha(u') \mid u' \in adj(u)\}\}$ és $m' = \{\{\alpha(v') \mid v' \in adj(f(u))\}\}$ multihalmazok esetén, ezért $\{\{\alpha(u') \mid u' \in adj(u)\}\} \preccurlyeq \{\{\alpha(v') \mid v' \in adj(f(u))\}\}$. Az állítás második fele triviálisan következik a $\preccurlyeq'$ definíciójából. □

3.12. Definíció *Legyen $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ tetszőleges részgráf-izomorf-konzisztens címkézés. Ezen ℓ alapú részgráf-izomorf-konzisztens címkézés-sorozat alatt az $\ell^0, \ell^1, \ell^2, \ldots$ sorozatot értjük, ahol*

$$\ell^i := \begin{cases} \ell, & \text{ha } i = 0 \\ \ell^{i-1} \text{ szomszédokra történő kiterjesztése} & \text{különben} \end{cases}$$

3.13. Tétel *Legyen $\ell = (\mathbb{L}, \preccurlyeq, \alpha)$ tetszőleges részgráf-izomorf-konzisztens címkézés, és tekintsük az ezen címkézés alapú ℓ^i címkézés-sorozatot. Ekkor*

i. az iteráció során a $D(x_u)$ halmazok nem bővülnek

ii. ha az i. iteráció során nem változnak a $D(x_u)$ halmazok, akkor semelyik későbbi iteráció során sem történik változás (ezen állapotot fixpontnak nevezzük)

iii. az algoritmus legfeljebb $n_q \cdot (n_t - 1) + 1$ lépés alatt fixpontot ér el.

Bizonyítás. Az állítás első két pontja ℓ^i és $\preccurlyeq^i$ definíciójából, a fentiekhez hasonló meggondolások alapján következik. Az állítás harmadik pontja pedig következik abból, hogy kezdetben legfeljebb $n_q \cdot n_t$ olyan (u, v) pár van, melyre $u \in V_q$ és $v \in D(x_u)$, továbbá egy tetszőleges iteráció során (esetleg az utolsó iteráció kivételével) ezen párok száma legalább eggyel csökken. Sőt, ha egy tetszőleges iteráció során valamely $D(x_u) = \emptyset$, akkor kész is, hiszen ekkor x_u-nak nem lehet értéket adni. □

A fenti gondolatmenetet írja le az ILF(k) algoritmus, mely még további két lépést hajt végre a futásidő csökkentése érdekében:

i. törli azon v target csúcsokat G_t-ből, melyek nem tartoznak a query csúcsok lehetséges képei közé, azaz $v \notin \bigcup_{u \in V_q} D(x_u)$ (ezáltal a későbbi iterációk során nem szükséges olyan csúcsok címkéjének meghatározása, melyekre nincs is szükség). Ezen lépés futásideje $O(n_q \cdot n_t)$.

ii. amennyiben $D(x_u) = \{v\}$, akkor egy új címkét, l_{uv}-t vezet be, mely az u és a v csúcsok címkéit jelöli és ezen címke csak önmagával kompatibilis (így a várható címke-összehasonlítások számát csökkenti). Ezen lépés futásideje $O(n_q)$.

3.14. Tétel *Az ILF(k) algoritmus futásideje $O(\min\{k, n_q \cdot n_t\} \cdot n_q \cdot n_t \cdot \Delta_q \cdot \Delta_t^{3/2})$.*

Bizonyítás. Az iterációk száma legfeljebb $\min\{k, n_q \cdot n_t\}$ a 3.13. tétel és az algoritmus k paramétere alapján, valamint korábban beláttuk, hogy egy iteráció műveletigénye $O(n_q \cdot n_t \cdot \Delta_q \cdot \Delta_t^{3/2})$. □

algorithm 3 ILF(k) algoritmus

Input : l^0 : kezdeti, részgráf-izomorf-konzisztens címkézés
k : felső korlát az iterációk számára

Output: „igaz" visszatérési érték esetén garantált, hogy $\forall u \in V_q\,(|D(x_u)| \geq 1)$, a „hamis" visszatérési érték az inkonzisztens állapotot jelöli

1: $i := 0$
2: **while** $i <= k$ **do**
3: A $D(x_u)$ halmazok módosítása l^i-nek megfelelően
4: **if** $D(x_u) = \emptyset$ valamely u query csúcsra **then**
5: **return** „hamis"
6: **end if**
7: **if** a módosítás során egyetlen halmaz mérete sem csökkent **then**
8: **return** „igaz" (az algoritmus fixpontba ért)
9: **end if**
10: Azon v target csúcsok törlése G_t-ből, melyekre $v \notin \cup_{u \in V_q} D(x_u)$
11: $D(x_u) = \{v\}$ esetén új címke, l_{uv} bevezetése, mely csak önmagával kompatibilis
12: ℓ^{i+1} meghatározása ℓ^i alapján
13: $i := i + 1$
14: **end while**
15: **return** „igaz"

3.4.3. LAD

A LAD (Local All Different) módszer az LV2002 és a GAC(AllDiff) módszerek általánosításának tekinthető, ugyanis LV2002-höz hasonlóan az $x_u := v$ értékadást megelőzően u és v lokális környezetére vonatkozó feltételek teljesülését ellenőrzi. Az LV2002 eljárástól eltérően nem a megfelelő szomszédsági halmazok számosságát hasonlítja össze, hanem egy páros gráfot épít ezen csúcsokból, majd megvizsgálja, hogy létezik-e az $adj(u)$ osztály csúcsait fedő párosítás ezen elkészített gráfban. Ezen, páros gráfot építő trükk található meg a GAC(AllDiff) módszernél is. Fontos különbség a két eljárás között, hogy a GAC(AllDiff) egyetlen páros gráfot határoz meg az adott G_q és G_t gráfokhoz, a LAD szűrési módszer valamennyi u query csúcs és v ($v \in D(x_u)$) targetcsúcs párhoz épít fel egy páros gráfot.

3.15. Definíció *Legyen u tetszőleges query csúcs és v ($v \in D(x_u)$) tetszőleges target csúcs. Az (u, v) párhoz tartozó $G_{(u,v)} = (adj(u), adj(v), E_{(u,v)})$ páros gráfot definiál-*

juk a következőképpen:

$$E_{(u,v)} := \{(u', v') \,|\, u' \in adj(u) \wedge v' \in adj(v) \wedge v' \in D(x_{u'}) \wedge l_q(u, u') = l_t(v, v))\}.$$

A részgráf-izomorfia probléma definíciójában (1.2.. definíció, 5. oldal) szereplő injektivitási feltétel alapján könnyen látható, hogy az $x_u = v$ értékadás kizárólag abban az esetben tartozhat valamely részgráf-izomorfizmushoz, ha $G_{(u,v)}$-ben létezik $adj(u)$-t fedő párosítás. Az eredeti cikkben a szerzők az eljárást címkézetlen gráfokra ismertették, melyet a molekulagráfok vizsgálata esetén kiegészíthettünk az (u, u') és (v, v') élek címkéinek összehasonlításával. Megjegyezzük, hogy az u' és a v' csúcsok esetén nem szükséges a csúcscímkék összehasonlítása, hiszen a $D(x_u)$ halmazok inicializálása során már vizsgáltuk ezen címkék egyezőségét.

Korábban említettük, hogy a $D(x_u)$ halmazok ritkítását követően szükséges a feltételek ismételt ellenőrzése, hiszen a törlés következtében korábban teljesülő kényszerek romolhattak el. A LAD-algoritmus ezen ellenőrzést végzi el a keresési fa tetszőleges csúcsában, melyet az alábbiakban ismertetünk. A $G_{(u,v)}$ páros gráf definíciójából könnyen adódik, hogy az (u, v) él kizárólag azon $G_{(u',v')}$ gráfokban szerepel, melyekre $u' \in adj(u) \wedge v' \in D(x_{u'}) \cap adj(v)$. Így, amennyiben v-t töröljük $D(x_u)$-ből ezen szűrési módszer alapján, akkor kizárólag ezen $G_{(u',v')}$ gráfokra szükséges újra vizsgálni, hogy a módosítást követően továbbra is tartalmaznak-e $adj(u')$-t fedő párosítást. Ezen gondolatmenetet írja le a LAD algoritmus, mely egy S halmazban tárolja azon (u, v) $(u \in V_q \wedge v \in D(x_u))$ párokat, amelyekhez tartozó $G_{(u,v)}$ páros gráfokat (újra) megvizsgálni szükséges. A visszalépéses keresés kezdőlépésében $S = \{(u, v) \,|\, u \in V_q \wedge v \in D(x_u)\}$; a keresés egy tetszőleges belső pontján kizárólag azon u query csúcsokhoz tartozó gráfokat vizsgálunk, melyek képe még nem ismert, azaz $|D(x_u)| > 1$, ezért ekkor $S = \{(u, v) \,|\, u \in V_q \wedge |D(x_u)| > 1 \wedge v \in D(x_u)\}$.

Az algoritmus visszatérési értéke egy logikai érték, mely azt reprezentálja, hogy valamennyi $D(x_u)$ halmaz tartalmaz legalább egy elemet, ellenkező esetben ezen állapotot tovább bővítve biztosan nem kaphatunk részgráf-izomorfizmust. Amennyiben visszatérési érték „igaz”, az algoritmus garantálja, hogy az összes (u, v) $(u \in V_q \wedge v \in D(x_u))$ párhoz tartozó $G_{(u,v)}$ gráf rendelkezik $adj(u)$-t fedő párosítással.

3.7. Példa

Tekintsük a 3.1. ábrán látható G_q és G_t gráfokat. Az (u_3, v_4) és a az (u_8, v_{14})-hez tartozó $G_{(u,v)}$ páros gráfok láthatóak a 3.3 ábrán. A $G_{(u_3,v_4)}$ páros gráfban sérül a Hall-feltétel az $\{u_2, u_8\}$ halmaz esetén, így ezen gráfban nem létezik $adj(u_3)$-at fedő párosítás. Fontos különbség a korábbi módszerekhez képest, hogy a LAD(AllDiff) már

algorithm 4 LAD algoritmus

Input : S, ahol $S \subseteq V_q \times V_t$, ahol az S-beli (u, v) párokra szükséges ellenőrizni, hogy tartalmaznak-e $adj(u)$-t fedő párosítást

Output: „igaz" visszatérési érték esetén garantált, hogy $\forall u \in V_q \forall v \in D(x_u)$ esetén létezik $G_{(u,v)}$-ben $adj(u)$-t fedő párosítás; a „hamis" visszatérési érték az inkonzisztens állapotot jelöli

```
1: while S ≠ ∅ do
2:    Legyen (u, v) egy tetszőleges S-beli elem
3:    S := S \ {(u, v)}
4:    if G_(u,v)-ben nem létezik adj(u)-t fedő párosítás then
5:       D(x_u) := D(x_u) \ {v}
6:       if D(x_u) = ∅ then
7:          return „hamis"
8:       end if
9:       S := S ∪ {(u', v') | u' ∈ adj(u) ∧ v' ∈ D(x_u') ∩ adj(v)}
10:   end if
11: end while
12: return „igaz"
```

az $x_3 := v_4$ értékadást sem engedi, míg a korábban tárgyalt módszerek az értékadást követően, a $D(x_u)$ halmazok ritkítása során kerültek inkonzisztens állapotba.

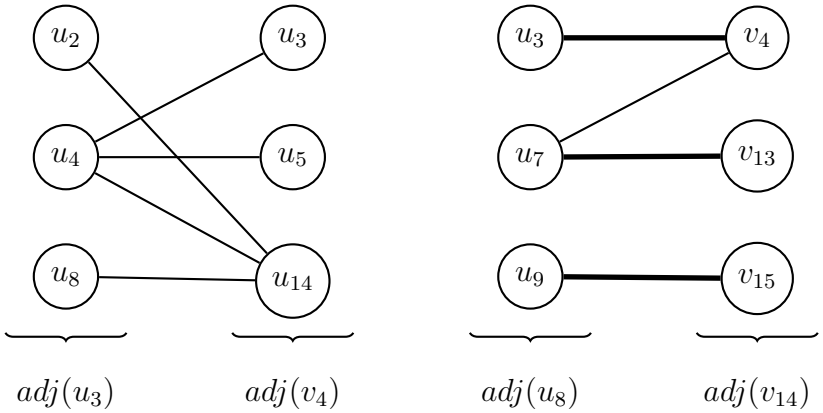
A $G_{(u_8,v_{14})}$ páros gráfban az $adj(u_8)$-at fedő párosítást vastag élekkel jelöltük, így v_{14} nem törölhető a LAD(AllDiff) szűrési módszer alapján $D(x_8)$-ból.

A LAD algoritmus a fenti gondolatmenet alapján először törli $D(x_3)$-ből v_4-et, majd újra vizsgálja például a $G_{(u_8,v_{14})}$ gráfot is. Ebben az esetben a 3.4. ábrán látható gráfot kapjuk, melyben az $u_3 \in adj(u_8)$ query csúcsra nem illeszkedik él, így $adj(u_8)$-t fedő párosítás sem létezik, ezáltal v_{14} törölhető $D(x_8)$-ból.

3.16. Tétel *A LAD eljárás szigorúan erősebb az LV2002 szűrési módszernél.*

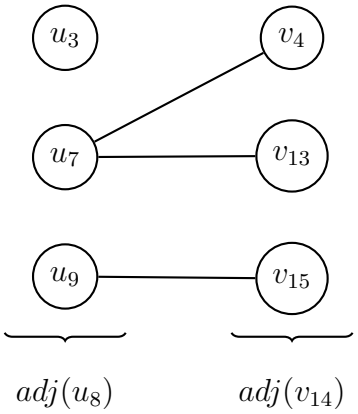
Bizonyítás. A Hall-feltétel teljesül az $adj(u)$ halmazra, ezért a LAD módszer legalább olyan erős, mint az LV2002. Az állítás azon része, hogy a LAD szigorúan erősebb az LV2002 módszernél, következik abból, hogy a bemutatott példa esetén a LAD inkonzisztenciát detektál az $x_3 := v_4$ esetén, míg az LV2002 módszer nem szűri ki ezen értékadást, mivel

$$\mathcal{F}(u_3, v_4) = \bigcup_{u' \in adj(u)} D(x_{u'}) \cap adj(v_4) = (D(x_3) \cup D(x_5) \cup D(x_{14})) \cap \{v_3, v_5, v_{14}\} =$$



(a) sérül a Hall-feltétel (b) van párosítás

3.3. ábra: A $G_{(u,v)}$ páros gráfok a 3.1. ábrán látható G_q és G_t gráfok esetén



3.4. ábra: A $G_{(u_8,v_{14})}$ páros gráf a $D(x_3) := D(x_3) \setminus \{v_4\}$ módosítást követően

$= \{v_3, v_5, v_{14}\}$ és $|\mathcal{F}(u_3, v_4)| = |adj(v_4)| = 3$. □

A LAD algoritmus és az Ullmann-algoritmusnál alkalmazott „finomítási kritérium" 3. pontja között némi hasonlóság figyelhető meg. Mindkét eljárás az $x_u := v$ értékadást megelőzően az u query csúcs és a v target csúcs szomszédainak lehetséges képeit vizsgálja meg. Fontos eltérés a két eljárás között, hogy az Ullmann-algoritmus a LAD algoritmussal ellentétben nem ellenőrzi, hogy u különböző szomszédjainak képe csak különböző target csúcsok lehetnek. A LAD algoritmus figyeli, hogy teljesülhet-e ezen injektivitási kritérium, valamint megmutatja, hogy a Hopcroft-Karp algoritmus hogyan alkalmazható ezen feltétel ellenőrzésére. További különbség, hogy az Ullmann-algoritmus csupán az u_{i+1} lehetséges képeit tartalmazó halmazból próbál meg elemeket törölni, a LAD eljárás ezzel szemben valamennyi $D(x_u)$ halmaz méretének csökkentésre tesz kísérletet.

3.17. Tétel *A LAD algoritmus futásideje* $O(n_q \cdot n_t \cdot \Delta_q^2 \cdot \Delta_t^2)$.

Bizonyítás. A Hopcroft-Karp algoritmus alkalmazásával $O(|E| \cdot \sqrt{|A \cup B|})$ idő alatt eldönthető, hogy a $G = (A, B, E)$ páros gráfban található-e A-t fedő párosítás [9, 23]. A $G_{(u,v)}$ páros gráf csúcsainak száma $|adj(u)| + |adj(v)| \leq 2 \cdot |adj(v)| \leq 2 \cdot \Delta_t$ ($|adj(u)| > |adj(v)|$ esetén $x_u := v$ biztosan nem lehetséges), éleinek száma pedig legfeljebb $|adj(u)| \cdot |adj(v)| \leq \Delta_q(u) \cdot \Delta_t(v)$; ezért annak vizsgálata, hogy a $G_{(u,v)}$ páros gráf tartalmaz-e $adj(u)$-t fedő párosítást, $O(\Delta_q \cdot \Delta_t^{3/2})$ idő alatt megválaszolható.

Annak érdekében, hogy a visszalépéses keresés lépései alkalmával a páros gráfok másolása elkerülhető legyen, a LAD algoritmus a visszalépéses keresés során valamennyi $G_{(u,v)}$ ($u \in V_q, v \in D(x_u)$) páros gráfot pontosan egyszer tárolja, majd az előrelépések és a visszalépések során módosítja az aktuális állapotnak megfelelően. Az algoritmus tárolja az ezen páros gráfokhoz tartozó, $adj(u)$-t fedő párosításokat is, melynek tárigénye $O(n_q \cdot n_t \cdot \Delta_q)$, mivel legfeljebb $n_q \cdot n_t$ páros gráf van és minden párosítás feljegyezhető $O(\Delta_q)$ memória felhasználásával. A $G_{(u,v)}$ páros gráfok száma legfeljebb $n_q \cdot n_t$, ezért a visszalépéses keresés kezdőlépésében a páros gráfokhoz tartozó párosítások meghatározhatóak $O(n_q \cdot n_t \cdot \Delta_q \cdot \Delta_t^{3/2})$ idő alatt.

Abban az esetben, ha a LAD algoritmus törli a v értéket a $D(x_u)$ halmazból, szükséges azon $G_{(u',v')}$ gráfok módosítására és esetleg új, $adj(u')$-t fedő párosítás meghatározására, melyre $u' \in adj(u) \wedge v' \in adj(v) \cap D(x_{u'})$. Ezen gráfok száma legrosszabb esetben $\Delta_q \cdot \Delta_t$. Az új párosítás a Hopcroft-Karp algoritmus alkalmazásával meghatározható $O(\Delta_q \cdot \Delta_t)$ idő alatt, felhasználva, hogy rendelkezésünkre áll a korábbi párosítás, melynek mérete eggyel kisebb az újonnan keresett párosítás méreténél. A fentiek alapján az új párosítások meghatározásának időigénye $O(\Delta_q^2 \cdot \Delta_t^2)$. A LAD algoritmus legrosszabb esetben egyetlen v' értéket töröl $D(x_{u'})$-ből, S mérete kezdetben pedig $O(n_q \cdot n_t)$, így az algoritmus futásideje $O(n_q \cdot n_t \cdot \Delta_q^2 \cdot \Delta_t^2)$.

4. fejezet

Összefoglalás

Korábbi kutatások során az NP-teljes részgráf-izomorfia feladatot megoldó Ullmann-algoritmussal, a VF2-vel, a QuickSI algoritmussal és néhány előszűrési módszerrel foglalkoztunk elméleti és gyakorlati szempontból. Jelen dolgozatban a problémát és a lehetséges megoldási módszereket egy más megközelítésből tekintettük át. Először a problémát CSP feladatként fogalmaztuk meg, majd általános CSP megoldási eljárásokat vizsgáltunk. Ezt követően speciális, a részgráf-izomorfiát eldöntő szűrési módszerekkel, az LV2002-vel, az ILF(k)-val és a LAD algoritmussal foglalkoztunk, melyeket példákon keresztül is szemléltettünk.

Az újonnan megismert módszerek tárgyalása során ezen eljárások ötleteit összehasonlítottuk a korábban tárgyalt algoritmusoknál alkalmazott heurisztikákkal. Számos hasonlóság fedezhető fel ezen módszerek között. Az LV2002 eljárás csupán kis mértékben tér el a VF2 által alkalmazott, a szomszédsági halmazok méretére vonatkozó feltétel ellenőrzésétől. Az ILF(k) eljárás a csúcsok egyre nagyobb sugarú környezetét tekinti, hasonlóan az ECFP generáláshoz. Fontos különbség az ILF(k) szűrési módszer és az ECFP között, hogy az ECFP nem alkalmazható a részgráf-izomorfia probléma esetén előszűrés céljából. A LAD algoritmus az Ullmann-algoritmushoz hasonlóan a csúcsok szomszédait hasonlítja össze, azonban a LAD algoritmus a szomszédok képeinek injektivitását is felveszi a kényszerek közé, majd megmutatja, hogy a Hopcroft-Karp párosítási algoritmus hogyan alkalmazható ezen esetben.

A vizsgált eljárásokat a szerzők címkézetlen gráfokra ismertették, melyeket kiegészítettünk úgy, hogy csúcs- és élcímkézett gráfok (pl. molekulagráfok) esetén is alkalmazhatóak legyen. A molekulagráfok esetén az egyes csúcs- és élcímkék eloszlása nem egyenletes, pl. egy molekula általában lényegesen több „C”, „N” és „O” atomot tartalmaz, mint például „F” vagy „Cl” atomot. Az algoritmusok módosításával ezen okok miatt várhatóan erősebb szűrési módszert is kapunk.

Irodalomjegyzék

[1] Markush structure. http://en.wikipedia.org/wiki/Markush_structure.

[2] Simplified molecular-input line-entry system. http://en.wikipedia.org/wiki/Simplified_molecular−input_line−entry_system.

[3] Gilles Audemard, Christophe Lecoutre, Mouny Samy-Modeliar, Gilles Goncalves, and Daniel Porumbel. Scoring-Based Neighborhood Dominance for the Subgraph Isomorphism Problem. In *Principles and Practice of Constraint Programming. 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 2014 of *Lecture Notes in Computer Science*. Springer, 2010.

[4] Nathan Brown. Chemoinformatics – an introduction for computer scientists. *ACM Computing Surveys*, pages 8:1–8:38, 2009.

[5] Horst Bunke, Pasquale Foggia, C. Guidobaldi, Carlo Sansone, and Mario Vento. A comparison of algorithms for maximum common subgraph on randomly connected graphs. In Terry Caelli, Adnan Amin, Robert P. W. Duin, Mohamed S. Kamel, and Dick de Ridder, editors, *SSPR/SPR*, volume 2396 of *Lecture Notes in Computer Science*, pages 123–132. Springer, 2002.

[6] Charles J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11(1):13–21, 1981.

[7] Donatello Conte, Pasquale Foggia, and Mario Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *Journal of Graph Algorithms and Applications*, 11(1):99–143, 2007.

[8] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An Improved Algorithm for Matching Large Graphs. *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*, pages 149–159, 2001.

[9] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[10] Thomas Engel. Basic overview of chemoinformatics. *Journal of Chemical Information and Modeling*, 46:2267–2277, 2006.

[11] Péter Englert and Péter Kovács. Efficient heuristics for maximum common substructure search. *Journal of Chemical Information and Modeling*, 55(0):941–955, 2015.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[13] Rosalba Giugno and Dennis Shasha. Graphgrep: A fast and universal method for querying graphs. In *16th International Conference on Pattern Recognition*, pages 112–115. IEEE Computer Society, 2002.

[14] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 172–184. ACM, 1974.

[15] Evgeny B. Krissinel and Kim Henrick. Common subgraph isomorphism detection by backtracking search. *Software: Practice and Experience*, 34(6):591–607, 2004.

[16] ChemAxon Ltd. Chemical hashed fingerprint. `https://docs.chemaxon.com/display/CD/Chemical+Hashed+Fingerprint`.

[17] ChemAxon Ltd. Extended Connectivity Fingerprint (ECFP). `https://docs.chemaxon.com/pages/viewpage.action?pageId=14483752`.

[18] George S. Lueker and Kellogg S. Booth. A linear time algorithm for deciding interval graph isomorphism. *Journal of ACM*, 26(2):183–195, 1979.

[19] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[20] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 –112, 2014. `http://pallini.di.uniroma1.it`.

[21] José Luis López Presa, Antonio Fernández Anta, and Luis Núñez Chiroque. Algorithm conauto for graph isomorphism testing and automorphism group computation. `https://sites.google.com/site/giconauto/`.

[22] John W. Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16:2002, 2002.

[23] Jean-Charles Régin. A Filtering Algorithm for Constraints of Difference in CSPs. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, pages 362–367. American Association for Artificial Intelligence, 1994.

[24] Haichuan Shang, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. Taming Verification Hardness: An Efficient Algorithm for Testing Subgraph Isomorphism. *Proceedings of the VLDB Endowment*, pages 364–375, 2008.

[25] Michael B. Smith. *March's Advanced Organic Chemistry: Reactions, Mechanisms, and Structure*. March's Advanced Organic Chemistry. Wiley, 2013.

[26] Christine Solnon. Solving Subgraph Isomorphism with an AllDifferent-based Filtering algorithm. `http://liris.cnrs.fr/csolnon/LAD.html`.

[27] Christine Solnon. AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174:850–864, 2010.

[28] Sébastien Sorlin and Christine Solnon. A Global Constraint for Graph Isomorphism Problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems . First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004. Proceedings*, volume 3011 2004 of *Lecture Notes in Computer Science*. Springer, 2004.

[29] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23:31–42, 1976.

[30] Mónika Vigula. Hatékony algoritmusok kémiai gráfok részgráf-izomorfia vizsgálatára, 2012. Eötvös Loránd Tudományegyetem, Informatikai Kar, Programtervező informatikus BSc szakdolgozat.

[31] Mónika Vigula. Hatékony részstruktúra-kereső algoritmusok a kémiai informatika területén, 2012. Eötvös Loránd Tudományegyetem, Informatikai Kar, Tudományos Diákköri Dolgozat.

[32] Mónika Vigula. A részgráf-izomorfia probléma adatbázisokban, 2012. Eötvös Loránd Tudományegyetem, Természettudományi Kar, Matematika BSc szakdolgozat.

[33] Mónika Vigula. Efficient implementation of algorithms for solving subgraph isomorphism problem in cheminformatics. 2013. Middle-European Conference on Applied Theoretical Computer Science (MATCOS) Koper, Slovenia, October 10th and 11th, 2013., `http://matcos.pint.upr.si/en/resources/files/program/vigula.pdf`.

[34] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, pages 31–36, 1988.

[35] Stéphane Zampelli, Yves Deville, and Christine Solnon. Solving Subgraph Isomorphism Problems with Constraint Programming. *Constraints*, 15:327–353, 2010.

[36] Stéphane Zampelli, Yves Deville, Christine Solnon, Sébastien Sorlin, and Pierre Dupont. Filtering for Subgraph Isomorphism. In *Principles and Practice of Constraint Programming – CP 2007 . 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007. Proceedings*, volume 4741 2007 of *Lecture Notes in Computer Science*. Springer, 2007.

[37] Shijie Zhang, Meng Hu, and Jiong Yang. Treepi: A novel graph indexing method. In *IEEE 23rd International Conference on In Data Engineering, 2007*, pages 966–975. IEEE, 2007.

[38] Peixiang Zhao. Graph indexing: Tree + Delta >= Graph. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07. VLDB Endowment, 2007.