A részgráf-izomorfia probléma adatbázisokban

BSc szakdolgozat

Témavezetők:

Dr. Tichler Krisztián
ELTE IK
Algoritmusok és Alkalmazásaik Tanszék
adjunktus

Dr. Fekete István
ELTE IK
Algoritmusok és Alkalmazásaik Tanszék
egyetemi docens

Készítette:

Vigula Mónika
Matematika BSc
Alkalmazott matematikus szakirány

Budapest, 2012

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg (a támogatás száma TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

Köszönetnyilvánítás

Köszönetet szeretnék mondani témavezetőimnek, dr. Tichler Krisztián Adjunktus Úrnak és dr. Fekete István Docens Úrnak.

Köszönöm dr. Tichler Krisztián Adjunktus Úrnak az értékes szakmai és stilisztikai tanácsait, építő javaslatait, melyek nélkülözhetetlenek voltak a téma részletes megértésében és szakdolgozatom elkészítésében.

Köszönöm dr. Fekete István Docens Úrnak, hogy megkedveltette velem az algoritmusok érdekes világát, a témafelvetést és a lehetőséget, hogy szakdolgozatom kutatási projekt keretében készíthettem el.

Köszönet illeti még Kovács Pétert (ChemAxon Kft.) a projekt során adott értékes tanácsaiért.

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

Tartalomjegyzék

Bevezetés

> „A tudomány, a technológia – ezt világosan és erősen akarom mondani – nem old meg minden problémát. De tudomány és technológia nélkül semmiféle problémát nem lehet megoldani." Teller Ede

A gráfok, mint adatmodellek a természettudományos és a mérnöki kutatások számos területén alkalmazhatóak. Számos esetben – pl.: gyógyszerkutatás, képfelismerés, számítógépes tanítás – szükség van különböző objektumok hasonlóságának eldöntésére. Amennyiben gráfot használunk az egyes objektumok reprezentálására, a feladatot visszavezethetjük gráfok közti hasonlóság eldöntésére.

Gráfizomorfia-vizsgálattal eldönthetjük, hogy két gráf azonos szerkezetű-e. A gráfizomorfizmus probléma biztosan NP-beli, azonban nem ismert, hogy P-beli vagy NP-teljes-e. Néhány gráftípusra léteznek polinomiális algoritmusok, például síkbarajzolható gráfokra [12], fokszámkorlátos gráfokra [18], intervallumgráfokra [17] és permutációgráfokra [7]. Legismertebb gráfizomorfiát eldöntő programcsomag a C nyelven implementált ingyenes nauty [19]. Szintén gráfizomorfiát vizsgáló algoritmus a conauto [16], mely a nauty-nál néhány esetben gyorsabb.

Hasonlóan vizsgálhatjuk, hogy két adott gráf, G_1 és G_2 esetén G_2 tartalmazza-e G_1-et. A részgráf-izomorfia-probléma NP-teljes. Részgráf-izomorfiát eldöntő eljárás a J. R. Ullmann 1976-ben publikált algoritmus [21], L. P. Cordella és mtsai. 2001-es algoritmusa, a VF2 [9], valamint a H. Shang és mtsai. által 2008-ban publikált algoritmus, a QuickSI [20].

Objektumok feldolgozása során szükség lehet két objektum hasonlóságának megállapítására. Ebben az esetben az objektumokat reprezentáló gráfok legnagyobb csúcsszámú közös részgráfját kell meghatároznunk. Két gráf legnagyobb közös részgráfjának megállapítása NP-teljes feladat [10]. Két gráf legnagyobb közös részgráfját meghatározó algoritmust találunk [6]-ban, [14]-ban és [8]-ban.

Jelen dolgozat a részgráf-izomorfia problémával foglalkozik, a gráfizomorfia- és a legnagyobb közös részgráf-problémákkal nem (bár az első speciális eset).

A részgráf-izomorfia problémát a kémiai informatika területéről merítettük, így példáink jelentős része molekulagráf. Egy egyszerű molekulát a következőképpen áb-

rázolhatunk gráfként. A gráf csúcsai legyenek az egyes atomok, és a gráf élei pedig legyenek az atomok közötti kötéseknek megfelelően. Molekulagráfok esetén nem foglalkozunk a molekulában szereplő hidrogénekkel és a hozzájuk kapcsolódó élekkel, hiszen ezeket kémiai tudásunk alapján meghatározhatjuk [5]. Megjegyezzük, hogy a szakdolgozatban szereplő példák esetén a kötések döntő többsége az egyszerűség kedvéért egyszeres kovalens kötés. Ez azonban nem jelent megszorítást a gráfokra vonatkozóan, hiszen az ismertetett eljárások más típusú kötések – illetve a kötéseknek megfelelő élek – esetén is alkalmazhatóak.

A kémiai informatikában (pl. gyógyszerkutatás során) gyakran merül fel az igény arra, hogy egy adott molekulagráf és molekulagráf-adatbázis esetén adjuk meg azon molekulagráfokat az adatbázisból, melyek részgráfként tartalmazzák az adott molekulagráfot. Mivel a részgráf-izomorfia-probléma NP-teljes, és az adatbázis gyakran több százezer, esetleg millió molekulagráfból áll, ezért azon megoldás, hogy az adatbázis minden egyes gráfjával végezzük el a részgráf-izomorfia-vizsgálatot, a gyakorlatban használhatatlan. Ezért az adatbázison előszűrést végzünk, ezáltal kiszűrve azon gráfokat az adatbázisból, melyeknek biztosan nem részgráfja az adott gráf. Csak az ezen eljárás után megmaradt adatbázisbeli gráfokkal végezzük el a részgráf-izomorfia-vizsgálatot.

A szakirodalomban több, különböző típusú eljárást találhatunk az előszűrésre. Az egyik leggyakoribb szűrési módszer a jellemző részgráfokkal történő előszűrés. Ebben az esetben a gráfadatbázis alapján meghatározott jellemző részgráfok segítségével végzünk szűrést az adatbázison. A jellemző részgráfok kiválasztását indexelésnek nevezzük. Több indexelési eljárást is ismerünk, melyek esetén a jellemző részgráfok különböző szerkezetűek. Út alapú indexelő módszer az R. Giugno és mtsai. publikált eljárása, a GraphGrep [11]. Shijie Zhang és mtsai. cikkében [23] a fa alapú részgráfokkal történő előszűrést vizsgálta. A P. Zhao és mtsai. által 2007-ben publikált algoritmus [24] a gyakori jellemző részfák mellett – amennyiben szükséges – nem fa alakú részgráfokat is kiválaszt az indexelés során.

A szakdolgozat a következőképpen épül fel. Az 1. fejezetben a definíciókat ismertetjük és a részgráf-izomorfia probléma NP-teljességét bizonyítjuk. Ezután két, részgráf-izomorfiát eldöntő algoritmust vizsgálunk: a 2. fejezetben J. R. Ullmann 1976-ben publikált algoritmusát [21] és a 3. fejezetben L. P. Cordella és mtsai. 2001-es algoritmusát, a VF2-t. Összehasonlítjuk ezen algoritmusok művelet- és memóriaigényét is. Megjegyezzük, hogy nem célunk meghatározni az összes részgráf-izomorfizmust a két gráf között. Abban az esetben, ha van részgráf-izomorfizmus a két gráf között, megadunk egyet, ellenkező esetben hamis visszatérési értékkel jelezzük, hogy a G_2 gráf nem tartalmazza a G_1 gráfot. Szükség esetén az Ullmann-

algoritmus, illetve a VF2 kis módosításával azonban az összes találat megadható. Mivel a problémát a kémiai informatika területéről merítettük, így példáink jelentős része molekula, habár az ismertetett algoritmusok más típusú gráfok esetén is alkalmazhatóak. Csak abban az esetben tekintünk el a kémiai gráf példától, ha könnyebben elmagyarázható az algoritmus egyszerű, címkézetlen gráf esetén.

Az Ullmann-algoritmus és a VF2 elemzése után a 4. fejezetben áttekintjük, hogy milyen módon kódolhatjuk el a gráfokat a hatékony részgráfizomorfia-vizsgálathoz. Ezen áttekintést H. Shang és mtsai. 2008-ban publikált cikke [20] alapján végezzük. Ehhez ismertetjük egy gráf egy sorozatként történő reprezentálását, majd egy újabb részgráf-izomorfiát eldöntő algoritmust ismertetünk, melynek inputjaként kapott G_1 gráf a bemutatott sorozatban van megadva.

Végül az általános szűrési eljárást, és az ezen módszer során felvetett problémának, a jellemző részgráfok kiválasztásának különböző módszereit tekintjük át az 5. fejezetben. Először egy út alapú indexelési módszert, a GraphGrep-et [11], majd a fa alapú jellemző részgráfok kiválasztását, a TreePi-t [23] mutatjuk be. Ezután P. Zhao és mtsai. 2007-ben publikált indexelési módszerét [24] vizsgáljuk. A különböző eljárásokat példákkal illusztráljuk.

1. fejezet

Alapfogalmak

1.1. Definíció *Adott két gráf, $G_1 = (V_1, E_1)$ és $G_2 = (V_2, E_2)$. A G_1 gráf részgráf-izomorf a G_2 gráffal, ha létezik olyan $f : V_1 \to V_2$ injektív függvény, melyre a következő teljesül:*

$$\forall u \in V_1 \forall v \in V_1 \ ((u, v) \in E_1 \Rightarrow (f(u), f(v)) \in E_2).$$

Jelölés: $G_1 \subseteq G_2$

Megjegyezzük, hogy címkézett (például kémiai) gráfok esetén további feltételek szükségesek.

1.2. Definíció *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.*

Ekkor a fenti definíció a következőképpen módosul.

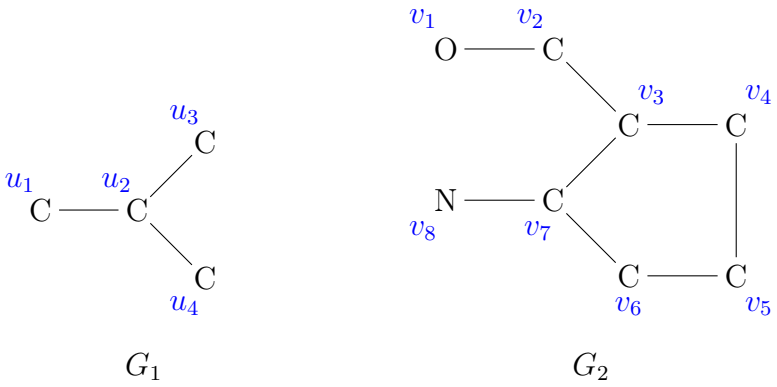
1.3. Definíció *Adott két címkézett gráf, $G_1 = (V_1, E_1, \Sigma_{V_1}, \Sigma_{E_1}, l_{V_1}, l_{E_1})$ és $G_2 = (V_2, E_2, \Sigma_{V_2}, \Sigma_{E_2}, l_{V_2}, l_{E_2})$. A G_1 gráf részgráf-izomorf a G_2 gráffal, ha van olyan $f : V_1 \to V_2$ injektív függvény, melyre a következő tulajdonságok teljesülnek:*

1. $\forall u, v \in V_1((u, v) \in E_1 \Rightarrow ((f(u), f(v)) \in E_2 \wedge l_{E_1}(u, v) = l_{E_2}(f(u), f(v))))$
2. $\forall u \in V_1 \ (l_{V_1}(u) = l_{V_2}(f(u)))$

1.1. Példa

Tekintsük az 1.1. ábrán látható G_1 és G_2 gráfokat. Egy lehetséges $f : V_1 \to V_2$:

$f(u_1) = v_2$
$f(u_2) = v_3$
$f(u_3) = v_4$
$f(u_4) = v_7$



1.1. ábra. Példa részgráf-izmorfiára

1.4. Tétel *Adott két gráf, $G_1 = (V_1, E_1)$ és $G_2 = (V_2, E_2)$. Annak eldöntése, hogy $G_1 \subseteq G_2$ NP-teljes feladat.*

Mielőtt a tételt bebizonyítanánk, szükségünk van néhány további definícióra.

1.5. Definíció *Klikknek nevezünk egy olyan részgráfot, amelynek bármely két különböző pontja között vezet él.*

1.6. Definíció *A klikk csúcsainak számát a klikk méretének nevezzük.*

1.7. Definíció *Klikk-probléma: adott egy G gráf és $k \in \mathbb{N}$ szám. Döntsük el, hogy G tartalmaz-e k méretű klikket.*

Az 1.4. tétel bizonyításához felhasználjuk az alábbi közismert tételt.

1.8. Tétel *A klikk-probléma NP-teljes.*

Bizonyítás: (1.4. tétel) Először bebizonyítjuk, hogy a részgráf-izomorfia-probléma NP-beli: a tanú az $f : V_1 \to V_2$ függvény. Polinom időben ellenőrizhető, hogy f kielégíti-e az 1.3. definícióban szereplő feltételeket. Végül a klikk-problémát, mely NP-teljes, polinomiálisan visszavezetjük a részgráf-izomorfia-problémára. Jelöljük (G, k)-val klikk-probléma egy inputját. Legyen G_1 a k pontú teljes gráf, $G_2 := G$. Ekkor G_1 pontosan akkor részgráf-izomorf G_2-vel, ha G tartalmaz k méretű klikket. □

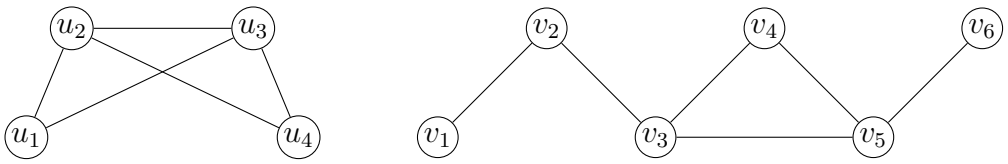
2. fejezet

Az Ullmann-algoritmus

A következőkben részgráf-izomorfia eldöntésére szolgáló algoritmusokat vizsgálunk. Először J. R. Ullmann 1976-os algoritmusát ismertetjük, mely a mai napig az egyik legtöbbször alkalmazott módszer. Ullmann 1976-os cikkében [21] egy visszalépéses keresésen alapuló algoritmust adott, mely eldönti, hogy $G_1 \subseteq G_2$ teljesüle. Az eljárás során egy ún. keresési fát építünk fel, melyben minden csúcs G_1 egy részgráfjának G_2 egy részgráfjára való leképezésnek felel meg. Célunk, hogy ezen leképezést bővítve, végül egy $f : V(G_1) \to V(G_2)$ injektív függvényt kapjunk, mely a csúcs- és az élcímkéket is megtartja. Ehhez a következőképpen járunk el. Minden $u_i \in V_1$ csúcshoz rendeljük hozzá a lehetséges képek halmazát: $\mathcal{H}_i := \{v_j | v_j \in V(G_2) \wedge deg(u_i) \leqslant deg(v_j) \wedge l_{V_1}(u_i) = l_{V_2}(v_j)\}$, ahol $deg(u_i)$ jelöli az u_i csúcs fokszámát.

2.1. Példa

Tekintsük a 2.1. ábrán látható példát. Az egyszerűség kedvéért tekintsünk el jelen esetben az élek és a csúcsok címkézésétől.



G_1 G_2

2.1. ábra. Példa az Ullmann-algoritmus inputjára

A $\mathcal{H}_i$ halmazokat meghatározva kapjuk:

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

Rögzítsünk G_1 csúcsainak egy tetszőleges sorrendjét: $u_1, u_2, \ldots, u_n$. Ekkor a keresési fát a következő gondolatmenettel építjük fel. A fa gyökere legyen az üreshalmaz. Ez lesz a 0. szint. A fa minden egyes csúcsa egy $f : V_1' \to V_2'$ leképezés, ahol $V_1' \subseteq V_1$ és $V_2' \subseteq V_2$. Nevezetesen, egy, az i. szinten lévő belső csúcsnak a következő felel meg:

$$f(u_1) = v_{j_1} \in \mathcal{H}_1$$

$$f(u_2) = v_{j_2} \in \mathcal{H}_2$$

$$\vdots$$

$$f(u_i) = v_{j_i} \in \mathcal{H}_i$$

Ezen leképezést kiterjesztjük u_{i+1}-re úgy, hogy $f(u_{i+1})$ lehetséges értékei a $\mathcal{H}_{i+1}$ halmazból valók legyenek. Így meghatároztuk azon $f : V_1 \to V_2$ leképezéseket, melyekre teljesül: $f(u_i) = v_{j_i} \in \mathcal{H}_i \ \forall\ i = 1, \ldots, n_1$, ahol $n_1 := |V(G_1)|$.

2.2. Példa

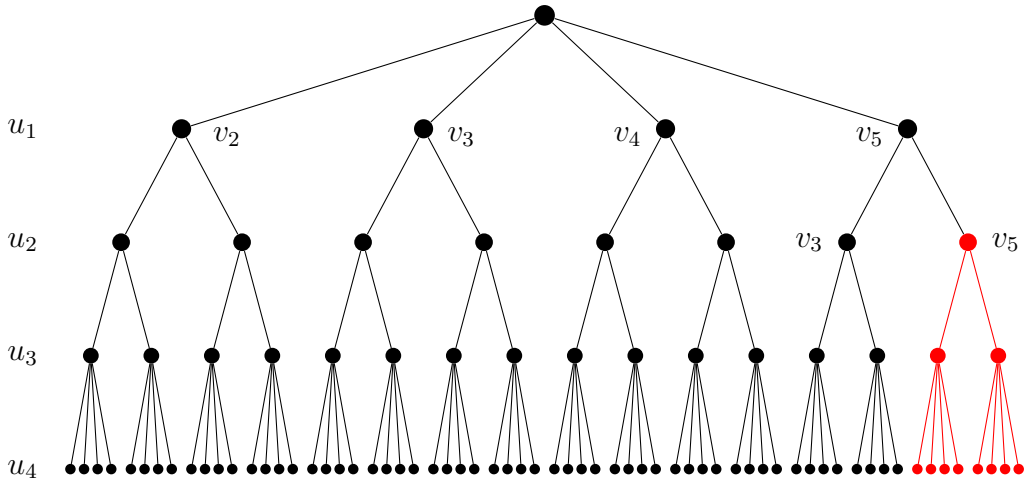
A 2.1. példa esetén a 2.2. ábrán látható keresési fát kapjuk, ha a csúcsokat az u_1, u_2, u_3, u_4 sorrendben dolgozzuk fel.

Célunk ezután, hogy ezen leképezések között keressünk egy olyan f-et, mely teljesíti a részgráf-izomorfizmus 1.3. definíciójában szereplő feltételeket, vagy hamis visszatérési értékkel jelezzük, amennyiben nincs ilyen f. Ezen f keresésére visszalépéses keresést használunk a kialakult fán. A teljes fa óriási méretű lehet, ezért nem építjük fel az egész fát explicit módon, csak azon részeit, melyeket be kell járnunk. Célunk, hogy a keresés során minél nagyobb részfákat egyben kihagyhassunk, és azon csúcsokat, melyek nem felelnek meg egy részgráf-izomorfizmusnak, minél hamarabb kiszűrjük, ezért ún. „Finomítási kritériumokat" vezetünk be.

Tekintsük a keresési fa egy belső csúcsát:

$$f(u_1) = v_{j_1} \in \mathcal{H}_1$$

$$f(u_2) = v_{j_2} \in \mathcal{H}_2$$



2.2. ábra. A keresési fa a 2.1. ábrán látható input esetén

$\vdots$

$f(u_i) = v_{j_i} \in \mathcal{H}_i$

Az u_{i+1} csúcshoz hozzárendelhetjük a $v \in \mathcal{H}_{i+1}$ csúcsot,

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

$$\forall m \in [1, \ldots, i]((u_m, u_{i+1}) \in E_1 \Rightarrow ((f(u_m), f(u_{i+1})) \in E_2 \wedge l_{E_1}(u_m, u_{i+1}) = l_{E_2}(v_{j_m}, v)))$$

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

Amennyiben ezen feltételek valamelyike nem teljesül, akkor töröljük v-t u_i lehetséges képei közül.

2.3. Példa

Az 1. *kritérium alapján a 2.1. példa esetén törölhetjük a pirossal jelölt csúcsokat a keresési fából, mivel ekkor* $f(u_1) = f(u_2) = v_5$*, ami ellentmond az injektivitásnak.*

A finomítási eljárás során folyamatosan figyeljük, hogy valamelyik $\mathcal{H}_i$ üres halmaze. Ezen esetben az algoritmus visszalép az előző szintre. Amennyiben az algoritmus a 0. szintről lépne vissza, az eljárást hamis visszatérési értékkel leállítjuk.

Az algoritmus implementálásához szükséges két tömb: $F = (F_1, \ldots, F_{n_1})$ és $H = (H_1, \ldots, H_{n_2})$, ahol $n_1 := |V(G_1)|$, $n_2 := |V(G_2)|$, illetve egy $M = (m_{ij})_{n_1 \times n_2}$ mátrix. H segítségével tároljuk, hogy G_2 mely csúcsait használtuk egy belső lépés

során, azaz $H_i = 1$, ha G_2 i. csúcsát megfeleltettük valamely $u_j \in V_1$ csúcsnak. A megfeleltetéseket az F vektorban tároljuk: $F_j = i$, ha $f(u_j) = v_i$. A keresés lehetséges lépéseit az M mátrix segítségével írjuk le. A mátrix a keresési fa egy szintjén a lehetséges elágazásokat tárolja. Az M mátrix segítségével a keresési fa méretét is csökkenthetjük, ahol inicializáláskor

$$m_{ij} := \begin{cases} 1 & \text{ha } deg(u_i) \leqslant deg(v_j) \wedge l_{V_1}(u_i) = l_{V_2}(v_j) \\ 0 & \text{különben} \end{cases}$$

Jelöljük d-vel, hogy a keresési fa épp mely szintjén járunk. Részgráf-izomorfizmust abban az esetben találunk, amennyiben minden V_1-beli csúcsnak megfeleltettünk egy V_2-beli csúcsot az 1.3. definíciónak megfelelően, azaz $d > n_1$. Az algoritmus $1-3$. sora ezen tulajdonságot ellenőrzi. A $4-5$. sor felsorolja azon fel nem használt csúcsokat, melyek kielégítik a Finomítási kritériumokat.

Finomítási kritériumok:

1. $l_{V_1}(u_d) = l_{V_2}(v_k)$
2. $\forall i \in [1, \ldots, n_2]((u_i, u_d) \in E(G_1) \Rightarrow (\exists j \in [1, \ldots, n_2])(v_j, v_k) \in E(G_2) \wedge m_{ij} = 1 \wedge l_{E_1}(u_i, u_d) = l_{E_2}(v_j, v_k))$

Abban az esetben, ha a Finomítási kritérium hamis az (u_d, v_k) párra, $m_{d,k}$-t 0-ra állítjuk, hiszen $f(u_d) = v_k$ biztosan nem lehetséges.

Az algoritmus fontos lépése az M mátrix ritkítása, melyet minden állapotmentést követően elvégzünk. Ekkor megvizsgáljuk, hogy ha m_{ij} igaz, akkor a G_1 gráf i. csúcsának a szomszédainak lehetséges képei között van-e olyan csúcs, mely a G_2 gráf j. csúcsának szomszédja. Amennyiben ez a feltétel nem teljesül, m_{ij}-t hamisra állítjuk. Ha sikerült a mátrix valamely elemét hamisra állítani, a mátrixot megpróbáljuk újra ritkítani.

Az algoritmus a 14. sorban visszalép az előző szintre, amennyiben a jelenlegi szinten nem talált részgráf-izomorfizmust.

Az algoritmus futása során fontos kérdés, hogy hogyan válasszuk meg a keresési sorrendet úgy, hogy minél hamarabb kiszűrjük azon csúcsokat a keresési fából, melyek biztosan nem felelnek meg egy részgráf-izomorfizmusnak G_1 és G_2 között. Ezen kérdést a későbbiek során részletesen megvizsgáljuk. Elöljáróban tekintsünk egy példát arra, hogy valóban érdemes a sorrendmegválasztás.

2.4. Példa

Legyen az Ullmann-algoritmus inputja a 2.3. ábrán látható gráfok. A $\mathcal{H}_i$ halmazokat meghatározva kapjuk:

Algoritmus 1. *UllmannAlgoritmus*(G_1, G_2, H, F, d)

Input : G_1, G_2 gráfok

F: n_1 hosszú vektor, kezdetben minden értéke 0

H: n_2 hosszú vektor, kezdetben minden értéke 0

d: keresési mélység, kezdetben $d := 0$

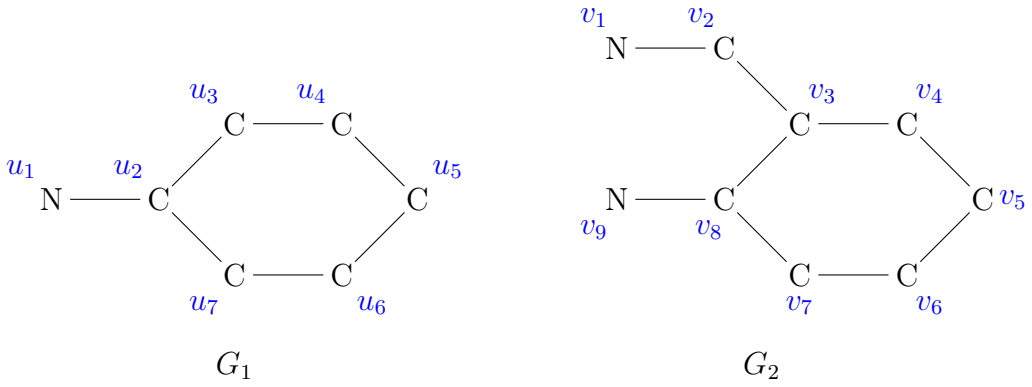
M: mátrix, mely korábban inicializálásra került

Output: Logikai: $G_1 \overset{?}{\subseteq} G_2$

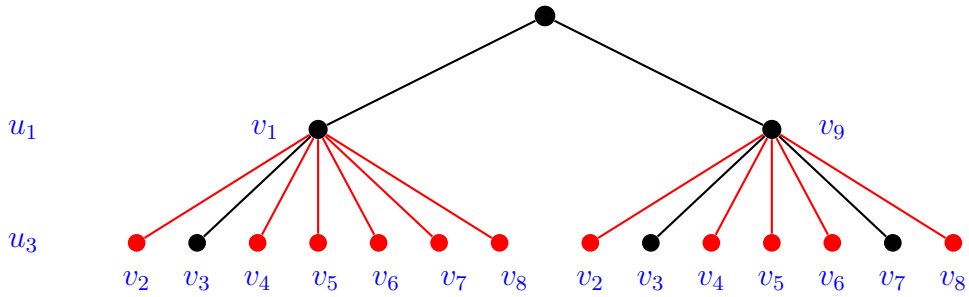
1: **if** $d > n_1$ **then**
2: **return** igaz
3: **end if**
4: **for each** $1 \leq j \leq n_2$ **do**
5: **if** Finomítási kritérium(d, j)=hamis **then**
6: Az állapot mentése és a mátrix ritkítása
7: **if** $UllmannAlgoritmus(G_1, G_2, H, F, d+1)$ **then**
8: **return** igaz
9: **else**
10: Előző állapot visszaállítása
11: **end if**
12: **end if**
13: **return** hamis
14: **end for each**

$$u_1 \mapsto \mathcal{H}_1 = \{v_1, v_9\}$$
$$u_2 \mapsto \mathcal{H}_2 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_3 \mapsto \mathcal{H}_3 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_4 \mapsto \mathcal{H}_4 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_5 \mapsto \mathcal{H}_5 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_6 \mapsto \mathcal{H}_6 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_7 \mapsto \mathcal{H}_7 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_8 \mapsto \mathcal{H}_8 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$u_9 \mapsto \mathcal{H}_9 = \{v_1, v_9\}$$

Tegyük fel, hogy a keresési fa 1. szintjét u_1, a 2. szintjét u_3 szerint építjük fel. Az így kapott keresési fa felső két szintjét láthatjuk a 2.4. ábrán. Vegyük észre, hogy $(u_1, u_2) \in E_1$ és $(u_2, u_3) \in E_1$, így szükséges, hogy $(f(u_1), f(u_2)) \in E_2$ és $(f(u_2), f(u_3)) \in E_2$ teljesüljön. Ezen észrevétel alapján törölhetünk a keresési fa 2. szintjén 11, pirossal jelölt csúcsot.



2.3. ábra. Példa az Ullmann-algoritmus inputjára



2.4. ábra. A keresési fa felső két szintje a 2.3. ábrán szereplő inputgráfok esetén

Az Ullmann-algoritmus memóriaigénye $O(N^3)$, ahol $N := n_1 + n_2$, mivel minden lépésben eltároljuk az M mátrixot, és a keresési fa konstrukciója alapján n_1 különböző lépés lehet.

3. fejezet

A VF2 algoritmus

Ezen fejezetben L. P. Cordella, P. Foggia, C. Sansone és M. Vento 2001-ben publikált algoritmusát [9], a VF2-t elemezzük. Az eredeti cikkben a részgráf-izomorfia-problémát a szerzők irányított gráfokra vizsgálták. Ezen algoritmust módosítottuk úgy, hogy irányítatlan gráfokra oldja meg a részgráf-izomorfia-problémát, mivel fő célunk a kémiai gráfok részgráf-izomorfia-vizsgálata, és ezen gráfok irányítatlanok.

Ezen módszer is, csakúgy mint a korábban ismertetett Ullmann-algoritmus, keresési fát épít fel. Ezen fán visszalépéses keresés segítségével keresünk részgráf-izomorfizmust (1.3. definíció, 4. oldal).

Az algoritmus fő gondolatmenete a következő: minden lépésben egy parciális izomorfizmust tekint G_1 és G_2 egy feszített részgráfja között, s ezen parciális izomorfizmust próbálja kibővíteni úgy, hogy végül részgráf-izomorfizmust kapjunk. Amennyiben nem lehetséges a kibővítés, visszalép az előző szintre.

Az algoritmus során egy állapotnak egy parciális izomorfizmus felel meg G_1 és G_2 egy feszített részgráfja között, mely kielégíti a részgráf-izomorfizmus definíciójában szereplő feltételeket. Jelöljük az egyes állapotokat s-sel, $M(s)$-sel pedig az s állapothoz tartozó azon (u_i, v_j) párokat, melyre $u_i \in V_1 \wedge v_j \in V_2 \wedge f(u_i) = v_j$. A kezdőállapot legyen s_0. Ekkor $M(s_0) = \emptyset$. Célunk ezen $M(s)$-t úgy kibővíteni, hogy végül részgráf-izomorfizmust kapjunk G_1 és G_2 között, vagy hamis visszatérési értékkel jelezni, amennyiben $G_1 \nsubseteq G_2$. Minden lépésben meghatározzuk azon jelölt (u, v) párokat, melyre $u \in V_1 \wedge v \in V_2$ és $M(s') := M(s) \cup \{(u, v)\}$ esetén továbbra is teljesülnek az 1.3. definícióban szereplő feltételek. Jelöljük $P(s)$-sel az s állapothoz tartozó jelölt párokat tartalmazó halmazt. P(s) minden egyes (u, v) elemére megvizsgáljuk, hogy $M(s)$-et (u, v)-vel bővítve továbbra is teljesülnek-e a definícióbeli tulajdonságok. Jelöljük $FF(s, u, v)$-vel (feasibility function) azon logikai függvényt, mely ezt a vizsgálatot elvégzi. Amennyiben $FF(s, u, v)$ értéke igaz, $M(s') := M(s) \cup \{(u, v)\}$-vel folytatjuk a vizsgálatot, ellenkező esetben $P(s)$ hal-

mazból töröljük az (u, v) párt. Ha $P(s) = \emptyset$, visszalép az előző szintre. Ha az eljárás s_0-ból is visszalép, hamis visszatérési értékkel leáll, mivel nincs részgráf-izomorfizmus G_1 és G_2 között.

Algoritmus 2. $Match(s)$

Input : egy s állapot; a kezdőállapot, s_0 esetén $M(s_0) = \emptyset$

Output: Logikai: $G_1 \overset{?}{\subseteq} G_2$

```
1: if M(s) G1 összes csúcsát lefedi then
2:    return igaz
3: else
4:    P(s) halmaz kiszámolása
5:    for each (u, v) ∈ P(s) do
6:       if FF(s,u,v)=igaz then
7:          M(s') := M(s) ∪ {(u, v)}
8:          Állapot mentése
9:          if Match(s') then
10:            return igaz
11:         else
12:            Előző állapot visszaállítása
13:         end if
14:      end if
15:   end for each
16:   return hamis
17: end if
```

A következőkben a $P(s)$ és az $FF(s, u, v)$ kiszámítását részletezzük. Annak kiküszöbölésére, hogy egy parciális izomorfizmust többször vizsgáljunk, feltesszük, hogy a V_1 teljesen rendezett halmaz. Jelöljük $G_1(s)$-sel, illetve $G_2(s)$-sel azon csúcsokat, melyeket $M(s)$ lefed G_1-ben, illetve G_2-ben. Ezen két halmaz egyértelműen meghatározott. Legyen $u_i \in G_1(s)$ egy tetszőleges csúcs. Jelölje $N(u_i)$ az u_i szomszédainak halmazát, azaz $N(u_i) := \{u | u \in V_1 \wedge (u_i, u) \in E_1\}$. Jelölje $N(G_1(s))$ a $G_1(s)$-beli csúcsok szomszédainak halmazát, azaz $N(G_1(s)) := \bigcup_{u_i \in G_1(s)} N(u_i)$. $N(G_2(s))$ hasonlóan definiálható.

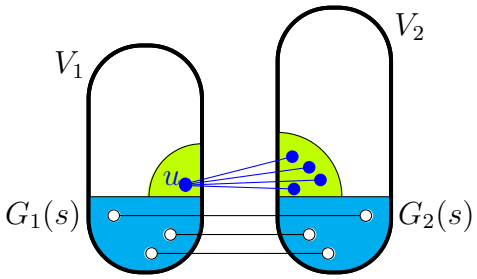
Az ábrán a következő színezést alkalmaztuk: kék színre szineztük a $G_1(s)$, illetve a $G_2(s)$ halmazt. Az $N(G_1(s))\backslash G_1(s)$ illetve $N(G_2(s))\backslash G_2(s)$ halmazt zöld szín jelöli.

A jelölt párokat tartalmazó halmaz, $P(s)$ kiszámításakor kétféle esetet különböztetünk meg $G_1(s)$ és $N(G_1(s))$ kapcsolata alapján.

1. eset: $N(G_1(s))\backslash G_1(s) \neq \emptyset$

   A keresési tér minimalizálása érdekében ekkor az $N(G_1(s))\backslash G_1(s)$ sorszám szerint minimális csúcsának – melyet jelöljünk u-val – keressük az izomorf képét, mely csak az $N(G_2(s))\backslash G_2(s)$ halmazból kerülhet ki, mivel u-nak van $G_1(s)$-beli szomszédja, így képének kell, hogy legyen $N(G_2(s))\backslash G_2(s)$-beli szomszédja az élstruktúra megtartása miatt. Így

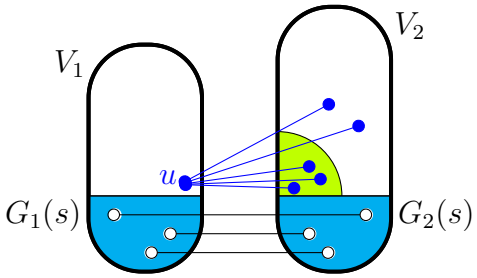
$$P(s) := \{\min(N(G_1(s))\backslash G_1(s))\} \times (N(G_2(s))\backslash G_2(s)).$$



3.1. ábra. 1. eset

2. eset: $N(G_1(s))\backslash G_1(s) = \emptyset$

   Amennyiben $s \neq s_0$, a gráf nem összefüggő. Az előző esethez hasonló gondolatmenettel adódik, hogy $P(s) := \{\min(V_1\backslash G_1(s))\} \times (V_2\backslash G_2(s))$.



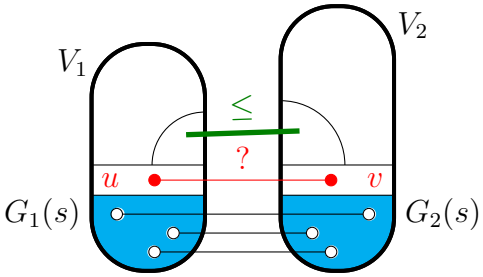
3.2. ábra. 2. eset

Az $FF(s, u, v)$ logikai függvény alkalmazásával a keresési fát csökkenthetjük. Ha értéke igaz, akkor garantált, hogy $s' := s \cup \{(u, v)\}$ is (parciális) izomorfizmus G_1 és G_2 között, amennyiben s az. $FF(s, u, v)$ kiszámításánál megvizsgáljuk, hogy u és v kompatibilis-e. Szükséges, hogy a csúcs- és az élcímkék egyezőek legyenek, illetve a 3.3. ábrán jelölt halmazok számosságára vonatkozó 3. feltételnek teljesülnie kell. Formálisan, a következő feltételeknek kell teljesülniük:

1. $l_{V_1}(u) = l_{V_2}(v)$

2. $\forall(u_i, v_j) \in M(s)\,((u, u_i) \in E_1 \Rightarrow ((v, v_j) \in E_2 \wedge l_{E_1}(u, u_i) = l_{E_2}(v, v_j)))$

3. $|N(G_1(s) \cup \{u\}) \backslash (G_1(s) \cup \{u\})| \leqslant |N(G_2(s) \cup \{v\}) \backslash (G_2(s) \cup \{v\})|$



3.3. ábra. $FF(s, u, v)$ 3. feltétel

A VF2 algoritmus implementálása során az alábbi tömböket használjuk: $F = (F_1, \ldots, F_{n_1})$, $H = (H_1, \ldots, H_{n_2})$, $S = (S_1, \ldots, S_{n_1})$ és $T = (T_1, \ldots, T_{n_2})$, ahol $n_1 := |V_1|$ és $n_2 := |V_2|$. F és H szerepe ugyanaz, mint az Ullmann-algoritmus esetén bemutatott tömböknek. F tárolja az aktuális parciális izomorfizmust: $F_i = v_j$, ha G_2 j. csúcsát feleltettük meg G_1 i. csúcsának, ebben az esetben $H_j = 1$. Az $S = (S_1, \ldots, S_{n_1})$ és $T = (T_1, \ldots, T_{n_2})$ tömbökben tároljuk az $G_1(s)$ és $N(G_1(s))$, illetve $G_2(s)$ és $N(G_2(s))$-beli csúcsokat. $S_i \neq 0$, ha $u_i \in G_1(s) \vee u_i \in N(G_1(s))$. Ebben az esetben S_i értéke azon keresési mélység, melynek során u_i bekerült a $G_1(s) \cup N(G_1(s))$ halmazba. T elemeinek értéke hasonlóan definiálható $G_2(s)$ és $N(G_2(s))$ alapján. Ezen vektorok mellett minden keresési mélység esetén nyilvántartjuk a keresési mélységet, $|N(G_1(s))\backslash G_1(s)|$ és $|N(G_2(s))\backslash G_s(s)|$ értékét, valamint azon (u, v) párt, mellyel kiegészítettük az s parciális izomorfizmust. Ezen adatszerkezetek alkalmazása mellett annak ellenőrzése, hogy $u_i \in N(G_1(s))\backslash G_1(s)$ konstans műveletigényű, mivel $F_i = 0$ és $S_i > 0$ ellenőrzése konstans időben megtehető. Így $P(s)$ kiszámítása legfeljebb $N := n_1 + n_2$-vel arányos, míg $FF(s, u, v)$ kiszámítása $N(u)$-val és $N(v)$-vel arányos.

A fenti tömbök fontos tulajdonsága, hogy amennyiben az i. tömbelem nem nulla az s állapot során, akkor az összes olyan állapot során, melyet s kibővítésével értünk el, továbbra sem lesz nulla. Így nem szükséges minden állapothoz külön tárolni ezen 4 tömböt. Abban az esetben, ha az algoritmus visszalép az előző szintre, mert nem tudja a parciális izomorfizmust tovább bővíteni, az állapothoz eltárolt konstans értékek alapján visszaállítja az előző állapotot.

Mivel a visszalépéses keresés során legfeljebb N állapot lehet, és minden állapothoz konstans méretű memóriát használunk, valamint a tömbök mérete is N-nel arányos, így a VF2 algoritmus memóriaigénye $O(N)$.

4. fejezet

QuickSI

Ebben a fejezetben egy újabb részgráf-izomorfiát eldöntő algoritmust vizsgálunk H. Shang, Y. Zhang, X. Lin és J. Xu Yu cikke [20] alapján. Először egy gráf egy lehetséges sorozatként történő reprezentációját, a QI-sorozatot részletezzük, majd egy újabb részgráf-izomorfiát eldöntő algoritmust, a QuickSI-módszert (quick subgraph isomorphism) vizsgálunk, melynek során a G_1 input gráf a QI-sorozattal van adva. Ezután heurisztika segítségével megmutatjuk, hogyan állítsuk elő egy gráf QI-sorozatát úgy, hogy a keresési fa méretét csökkenthessük.

A fejezet végén bemutatunk egy heurisztika megoldást az Ullmann-algoritmusnál említett kérdésre, hogy mily módon határozzuk meg a csúcsok feldolgozási sorrendjét úgy, hogy a keresési fa mérete várhatóan minél jobban csökkenjen, a QI-sorozat segítségével a fejezet végén válaszoljuk meg (azaz miképpen válasszunk a lehetséges QI-sorozatok közül).

4.1. QI-sorozat

Ezen alfejezetben egy gráf egy lehetséges sorozattá történő alakítását mutatjuk be, melyet a későbbiek során, egy újabb részgráf-izomorfiát meghatározó algoritmus bemutatása esetén felhasználunk.

4.1. Definíció *QI-sorozat: Adott egy n csúcsú G gráf. Ezen gráf egy lehetséges QI-sorozatán az alábbi kifejezést értjük:*

$$QI_G = T_1 R_{11} \dots R_{1i_1} T_2 R_{21} \dots R_{2i_2} \dots T_n R_{n1} \dots R_{ni_n},$$

mely G egy feszítőfáját határozza meg. T_i számos információt tárol:

1. *$T_i.v$ egy $u_j \in V(G)$ csúcsot*

2. *$[T_i.p, T_i.l]$ egy párt, ahol $T_i.p$ a $T_i.v$ csúcs szülőjét a feszítőfában, $T_i.l$ pedig a $T_i.v$ csúcs címkéjét jelöli*

Fontos megemlíteni, hogy a $T_1.v, T_2.v, \ldots, T_n.v$ határozza meg a csúcsok feldolgozási sorrendjét ($\bigcup_{i=1}^{n} T_i.v = V(G)$). Mivel a QI-sorozat feszítőfát határoz meg G-ben, így az esetleges fokszámfeltételt vagy a feszítőfában nem szereplő éleket is tárolnunk kell. Ezen extra információkat R_{ij} segítségével tároljuk. A fokszámfeltételt $[deg : d]$ formában tároljuk, ahol d a $T_i.v$ csúcs fokszáma a G gráfban. Annak kiküszöbölésére, hogy felesleges információt ne jegyezzünk fel, $d \leq 2$ esetén nem szerepel a fokszámfeltétel. Extra élt – olyan él, mely nem szerepel a feszítőfában – $[edge : k]$ formában tüntetünk fel, amennyiben $(T_i.v, T_k.v) \in E(G)$ és $i > k$. R_{ij} indexében j az extra feltétel sorszámát jelöli. Megjegyezzük, hogy nem feltétlen tartozik minden $T_i.v$ csúcshoz extra információ.

Egy gráfnak több QI-sorozata is lehet, de egy QI-sorozat egyértelműen meghatározza a gráfot.

4.1. Példa

Legyen G_1 a 4.2. ábrán szereplő gráf. G_1 két lehetséges QI-sorozatát mutatja be a 4.1. ábra. Vegyük észre, hogy az 1. QI-sorozat a csúcscímkék előfordulási száma alapján növekvően rendezett, mivel az N csúcscímkéjű csúcsok száma kisebb, mint a C címkéjű csúcsok száma; míg a 2. QI-sorozat nem ilyen.

Mivel két különböző QI-sorozat esetén a keresési fa mérete különböző, ezért a későbbiek során megvizsgáljuk, hogy hogyan határozzunk meg egy adott gráf hatékony QI-sorozatát.

4.2. Tétel *Legyen adott két gráf, G_1 és G_2. Tegyük fel, hogy QI_{G_1} és QI_{G_2} megegyezik. Ekkor G_1 és G_2 is megegyezik.*

Bizonyítás: QI_{G_1} egyértelműen meghatározza G_1-t, hasonlóan QI_{G_2} egyértelműen meghatározza G_2-t. Mivel a két QI-sorozat megegyezik, így G_1 és G_2 is. □

4.2. QuickSI-algoritmus

Ebben az alfejezetben a QuickSI (quick subgraph isomorphism) algoritmust ismertetjük. Legyen G_1 és G_2 a két input gráf, eldöntendő, hogy $G_1 \subseteq G_2$. A QuickSI algoritmus segítségével megvizsgáljuk, hogy G_2-nek van-e olyan G_2' részgráfja, melyre

$Type$	$[T_i.p, T_i.l]$	$T_i.v$
T_1	$[0, N]$	u_1
T_2	$[1, C]$	u_2
R_{21}	$[deg : 3]$	
T_3	$[2, C]$	u_3
T_4	$[3, C]$	u_4
T_5	$[4, C]$	u_5
T_6	$[5, C]$	u_6
T_7	$[6, C]$	u_7
R_{71}	$[edge : 2]$	

$Type$	$[T_i.p, T_i.l]$	$T_i.v$
T_1	$[0, C]$	u_4
T_2	$[1, C]$	u_5
T_3	$[1, C]$	u_3
T_4	$[2, C]$	u_6
T_5	$[4, C]$	u_7
T_6	$[5, C]$	u_2
R_{61}	$[deg : 3]$	
R_{62}	$[edge : 3]$	
T_7	$[6, N]$	u_1

4.1. ábra. G két lehetséges QI-sorozata

QI_{G_1} és $QI_{G'_2}$ megegyezik. Ekkor a 4.2. tétel alapján $G_1 \subseteq G_2$ teljesül. Az algoritmus a következő inputot várja:

1. QI_{G_1}-t
2. G_2 gráfot
3. $F = (F_1, \ldots, F_{n_1})$ és $H = (H_1, \ldots, H_{n_2})$ tömböt, ahol F és H minden értéke 0
4. keresési fa aktuális szintjét, értéke kezdetben 0

Hasonlóképpen, mint az Ulmann-algoritmus és VF2 esetén, az F tömb segítségével tároljuk az aktuális parciális izomorfizmust: $F_i = v_j$, ha a $T_i.v$ csúcsot megfeleltettük a $v_j \in V_2$ csúcsnak. A $H = (H_1, H_2, \ldots, H_{n_2})$ bináris tömb segítségével tároljuk, hogy G_2 mely csúcsait feleltettük meg valamely $u_i \in V_1$ csúcsnak. Tegyük fel, hogy $F_1, F_2, \ldots, F_i$ egy parciális izomorfizmus G_1 és G_2 között. Ekkor a QI-sorozatban szereplő esetleges R_{ij} $[deg : d]$ fokszámfeltétel esetén $deg_{G_2}(F_i) \geq d$, míg $R_{ij} =$ $[edge : k]$ esetén $(F_i, F_k) \in E_2$ teljesül.

Az algoritmus 1-3. sora a keresési mélység vizsgálatával ellenőrzi, hogy részgráf-izomorfizmust találtunk-e G_1 és G_2 között. Ha $d > n_1$, akkor G_2-nek találtunk olyan G'_2 részgráfját, melynek QI-sorozata megegyezik G_1 QI-sorozatával. Ekkor a 4.2. tétel következtében $G_1 \subseteq G_2$, így az algoritmus igaz visszatérési értékkel terminál. Ellenkező esetben a $T_d.v$ csúcsnak olyan képét keresi, melyre teljesülnek az esetleges fokszám- és visszakötő él feltételek, valamint a csúcs- és élcímkék is megegyeznek. Amennyiben nem talál megfelelő képet $T_d.v$-nek, visszalép az előző szintre. Abban az esetben, ha az összes esetet végigtekintve nem talált részgráf-izomorfizmust, hamis visszatérési értékkel terminál.

4.2. Példa

Tekintsük a következő példát. Legyen G_1 a 4.2. ábrán szereplő gráf, G_2 az adatbázisbeli G_2 gráf. A QuickSI algoritmus észreveszi, hogy v_1-t megfeleltetheti $T_1.v$-nek, így $F_1 := v_1$ és $H_1 := 1$ módosítással tárolja a parciális izomorfizmust. Ekkor $V := \{v_2\}$. $l(u_2) = T_2.l$, de a fokszámfeltétel, $[deg : 3]$ ellenőrzése során észreveszi, hogy $deg_{G_2}(v_2) = 2 \ngeq 3$, így H_1-et 0-ra módosítja és visszalép az előző szintre. $T_1.v$-t más G_2-beli csúcsnak, v_9-nek felelteti meg. Végül az algoritmus talál egy részgráf-izomorfiát G_1 és G_2 között: $F = [v_9, v_8, v_7, v_6, v_5, v_4, v_3]$-t vagy $F = [v_9, v_8, v_3, v_4, v_5, v_6, v_7]$-t.

A 3. algoritmus műveletigényének elemzéséhez vezessük be az alábbi fogalmakat.

4.3. Definíció *Adott a 3. algoritmus inputja, QI_{G_1} és G_2. Jelöljük B_j-vel a $QI^j_{G_1}$ és G_2 közötti parciális izomorfizmusok számát, ahol $QI^j_{G_1}$ jelöli a QI_{G_1} első j csúcsát tartalmazó kezdőszeletét: $QI^j_G = T_1R_{11} \dots R_{1i_1}T_2R_{21} \dots R_{2i_2} \dots T_jR_{j1} \dots R_{ji_j}$.*

Ekkor a 3. algoritmus műveletigényének, $T(QI_{G_1}, G_2)$-nek a nagyságrendje a következőképpen számítható. Mivel minden V_2-beli csúcsra előre kiszámíthatjuk a fokszámot, így a $QI^j_{G_1}$ és G_2 közti parciális izomorfizmus bővítése során a G_1 QI-sorozatában szereplő esetleges fokszámfeltétel teljesülését $O(1)$ idő alatt ellenőrizhetjük G_2-ben. Hasonlóan, ha G_2-t csúcsmátrix segítségével tároljuk, a visszakötő élre vonatkozó feltételnek az ellenőrzése $O(1)$ időben elvégezhető. A parciális izomorfizmus bővítéséhez szükségünk van azon csúcsok meghatározására, melyek $F_{T_j.p}$-nek szomszédja G_2-ben, és még nem feleltettük meg valamely G_1-beli csúcsnak. Ezen csúcsokat tartalmazó halmazt $O(deg_{G_2}(F_{T_j.p}))$ idő alatt határozhatjuk meg. Kezdetben $T_1.v$ lehetséges képeinek halmazát úgy kaphatjuk meg, ha $\forall u \in G_2$-beli csúcsra ellenőrizzük, hogy $T_1.v$ és u csúcscímkéje megegyezik-e. Ezen művelet $\Theta(n_2)$ időt igényel. Ezen gondolatmenet alapján $T(QI_{G_1}, G_2)$ kiszámítására a (4.1) egyenlőtlenséget kapjuk, melyet felülről becsülünk (4.2). Megjegyezzük, hogy $T(QI_{G_1}, G_2)$ pontos értéket nem tudjuk meghatározni, mivel a 3. algoritmus leáll abban az esetben, ha talált egy részgráf-izomorfizmust G_1 és G_2 között.

$$T(QI_{G_1}, G_2) = O\left(n_2 + B_1 \cdot r_1 + \sum_{k=1}^{n_1-1} \sum_{l=1}^{B_k} deg(k,l) \cdot r_{k+1}\right) \leq \tag{4.1}$$

$$O\left(n_2 + B_1 \cdot r_1 + \sum_{k=2}^{n_1} B_k \cdot deg_{max}(G_2) \cdot r_{k+1}\right) \tag{4.2}$$

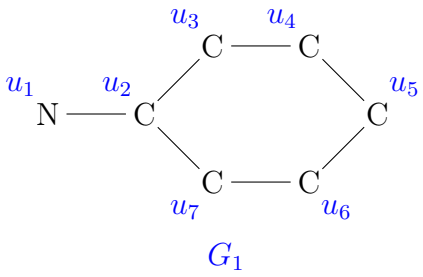
Ezen egyenlőtlenségek esetén r_l jelöli annak a vizsgálatnak a költségét, hogy a parciális izomorfizmus bővítése során $T_l.v$ és képének csúcscímkéje azonosak-e, illetve

Algoritmus 3. $QuickSI(QI_{G_1}, G_2, F, H, d)$

Input : QI_{G_1}: G_1 gráf QI-sorozata
$G_2 = (V_2, E_2)$ gráf
F: n_1 hosszú tömb, kezdetben minden értéke 0
H: n_2 hosszú tömb, kezdetben minden értéke 0
d : keresési mélység, értéke kezdetben 1

Output: Logikai: $G_1 \overset{?}{\subseteq} G_2$

```
1: if d > n_1 then
2:    return igaz
3: end if
4: T := T_d
5: V := ∅
6: if d = 1 then
7:    V := {v|v ∈ V_2 ∧ l(v) = T.l ∧ H_v = 0}
8: else
9:    V := {v|v ∈ V_2 ∧ (v, F_{T.p}) ∈ E_2 ∧ l(v) = T.l ∧ H_v = 0}
10: end if
11: for each v ∈ V do
12:    for each R_dj ∈ QI_{G_1} do
13:       if R_dj nem teljesül then
14:          goto 11. sor
15:       end if
16:    end for each
17:    F_d := v
18:    H_v := 1
19:    if QuickSI(QI_{G_1}, G_2, F, H, d + 1) then
20:       return igaz
21:    end if
22:    H_v := 0
23: end for each
24: return hamis
```



4.2. ábra. G_1

teljesülnek-e a $T_l.v$ csúcshoz tartozó extra feltételek, azaz $r_l := 1+|\{R_{il}|R_{il} \in QI_{G_1}\}|$.

A 3. algoritmus memóriaigénye $\Theta(|QI_{G_1}| + |G_2|)$, ahol $|G_2|$ jelöli a G_2 gráf tárolásához szükséges memóriát.

4.3. Hatékony QI-sorozat kiválasztása

A hatékony QI-sorozat kiválasztását [20] alapján határozzuk meg. Mint már korábban említettük, egy gráfnak több QI-sorozata lehet, s ezen QI-sorozatok esetén a részgráf-izomorfia-vizsgálat költsége különböző. Tekintsük erre a következő példát.

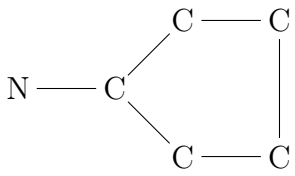
4.3. Példa

Legyen G_1 a 4.2. ábrán látható gráf, melynek két QI-sorozata szerepel a 4.1. ábrán, G_2 pedig legyen a 4.3. ábrán látható gráf. Ekkor az első QI-sorozat alapján a részgráf-izomorfia-vizsgálat költsége $T(QI_{G_1}, G_2) \leq 37$, míg a 2. QI-sorozat alapján $T(QI_{G_1}, G_2) \leq 161$.

A $T(QI_{G_1}, G_2)$-re vonatkozó (4.2) egyenlőtlenség alapján B_j minimimalizálásával a részgráf-izomorfia-vizsgálat költsége is csökken, azonban G_1 optimális QI-sorozatának meghatározása túl költséges ahhoz, hogy B_j-t minimalizálhassuk. Ehelyett B_j-t heurisztika segítségével próbáljuk meg csökkenteni, melyhez szükségünk van néhány definícióra.

4.4. Definíció *Adott egy x csúcscímke. Definiáljuk x átlagos belső tartóját, $\phi(x)$-et a következőképpen:* $\phi(x) := \dfrac{|\{u|\exists G' \in D(u \in V(G') \wedge l_{V_{G'}}(u) = x)\}|}{|\{G'|G' \in D \wedge \exists u \in V(G')(l_{V_{G'}}(u) = x)\}|}$.

Például a $\phi(C)$ az az érték, melyet úgy kapunk, ha C-atomok számát D-ben elosztjuk azon gráfok számával, melyek tartalmaznak C-t. Hasonlóan definiálhatjuk $\phi(y)$-t, ahol y egy tetszőleges élcímke.



G_2

4.3. ábra. G_2

4.5. Definíció *Adott egy y élcímke. Definiáljuk y átlagos belső tartóját, $\phi(y)$-t a következőképpen:* $\phi(y) := \frac{|\{e|\exists G' \in D(e \in E(G') \wedge l_{E_{G'}}(v) = y)\}|}{|\{G'|G' \in D \wedge \exists e \in E(G')(l_{E_{G'}}(e) = y)\}|}$.

4.6. Definíció *Adott a G_1 gráf és egy $u \in V_1$ csúcs. Definiáljuk u átlagos belső tartóját, $\phi(u)$-t a következőképpen:* $\phi(u) := \phi(l_{V_1}(u))$.

4.7. Definíció *Adott a G_1 gráf és egy $e \in E_1$ él. Definiáljuk e átlagos belső tartóját, $\phi(e)$-t a következőképpen:* $\phi(e) := \phi(l_{E_{G_1}}(e))$.

G_1 hatékony QI-sorozatának meghatározásához számítsuk ki minden G_1-beli csúcsra és élre az átlagos belső tartót. Készítsük el ezen értékek alapján G_w súlyozott gráfot G_1-ből úgy, hogy minden csúcshoz és élhez rendeljük hozzá az átlagos belső tartóját.

4.4. Példa

Tegyük fel, hogy a 4.2. ábrán látható G_1 gráf esetén a 4.4. ábrán szereplő értékeket kaptuk. Ezen értékek alapján kapott G_w gráf a 4.5. ábrán szerepel. A könnyebb áttekinthetőség érdekében a csúcsok és az élek átlagos belső tartóját különböző szín jelöli.

x	$\phi(x)$
N	1.5
C	6.1

y	$\phi(y)$
$N-C$	1.4
$C-C$	5.6

4.4. ábra. Csúcs- és élcímkék belső tartója

Ezután a G_w egy minimális feszítőfája alapján készítjük el a G_1 gráf QI-sorozatát. Egy minimális feszítőfa meghatározásához a Prim-algoritmust [13] használjuk, melyet kiegészítünk azon esetekre vonatkozó élválasztással, melyek esetén a Prim-algoritmus véletlenszerűen választana egy élt a minimális súlyú élek közül. Jelölje

T az aktuális feszítőfát G_1-ben, V_T és E_T pedig a feszítőfához tartozó csúcs- és élhalmazt. Legyen P azon jelölt éleket tartalmazó halmaz, melyek közül kiválasztjuk a következő lépésben a feszítőfabeli élt. Az első lépésben kiválasztjuk azon G_w-beli élt, melynek az átlagos belső tartója a legkisebb. Abban az esetben, ha több ilyen él is van, akkor azt az élt, mely csúcsainak fokszámösszege minimális, egyéb esetben véletlenszerűen választunk. Az első feszítőfabeli él, (u, v) kiválasztása után meghatározzuk, hogy mely csúcs legyen $T_1.v$ és $T_2.v$. Ha $\phi(u) \neq \phi(v)$, akkor $T_1.v$ legyen az a csúcs u és v közül, melynek az átlagos belső tartója minimális. Ellenkező esetben $T_i.v$ legyen u és v közül az csúcs, melynek fokszáma nagyobb. Abban az esetben, ha a fokszámok is megegyeznek, véletlenszerűen válasszunk.

Az élek és a csúcsok közti sorrend megválasztását az indokolja, hogy korábban határozzuk meg a ritkábban előforduló - pl.: N, O - csúcsok képét, s csak ezután keressük a gyakori csúcsok - kémiai gráfok esetén pl.: C - képét.

Az Élkiválasztás(G_w, P) algoritmus azzal az esettel foglalkozik, mikor több olyan minimális súlyú él van, melyek közül választjuk ki a következő feszítőfabeli élt. Ekkor válasszuk azt élt, mely az új csúccsal a legnagyobb részgráfot feszíti G_1-ben. Amennyiben több ilyen csúcs van, akkor azt, amelynek a fokszáma a legkisebb, egyébként válasszuk ki a csúcsot véletlenszerűen.

4.5. Példa

G_1 legyen a 4.2. ábrán látható gráf, melyből az átlagos belső tartók kiszámolása után a 4.5. ábrán szereplő gráfot kapjuk. Kezdetben az (u_1, u_2) él esetén minimális az átlagos belső tartó, így a T feszítőfához hozzáadjuk az (u_1, u_2) élt. Ezután $P = \{(u_2, u_3), (u_2, u_7)\}$. Mivel u_3 és u_7 fokszáma megegyezik, továbbá a feszített részgráfok is azonos méretűek, így véletlenszerűen választunk u_3 és u_7 közül. Legyen u_3 a kiválasztott csúcs. Tegyük fel, hogy ezen gondolatmenetet folytatva már az $(u_3, u_4), (u_4, u_5), (u_5, u_6)$ és (u_6, u_7) éleket is hozzáadtuk a feszítőfához. Ekkor az (u_2, u_7) él extra élként szerepel G_1 QI-sorozatában.

4.4. Algoritmusok összehasonlítása

Az Ullmann- és a QuickSI-algoritmust a [20] cikk alapján hasonlítjuk össze. Az Ullmann-algoritmus és a QuickSI-algoritmus futásidejét az AIDS Antiviral adatbázison tesztelve hasonlítjuk össze. Utóbbi esetén külön megvizsgáljuk, hogy mennyire befolyásolja a G_1 gráf egy véletlen és egy hatékony QI-sorozata az eredményt. Az általános adatbázis, melyet algoritmusok hatékonyságának megállapításához hasz-

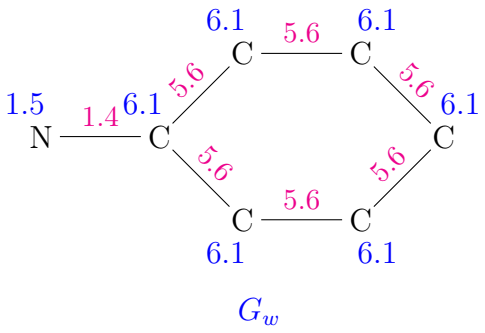
Algoritmus 4. Minimálisfeszítőfa(G_w)

Input : G_w: G_1-ből az átlagos belső tartó alapján elkészített súlyozott gráf

Output: T: G_w egy minimális feszítőfája

QI_{G_1}: G_1 egy QI-sorozata

1: $V_T := \emptyset$
2: $E_T := \emptyset$
3: $P := \{(u,v)|(u,v) \in E(G_w) \wedge \forall (u',v') \in E(G_w)(\phi(u,v) \leq \phi(u',v'))\}$
4: **if** $|P| > 1$ **then**
5: $P' := \{(u,v)|(u,v) \in P \wedge \forall (u',v') \in P(deg(u) + deg(v) \leq deg(u') + deg(v'))\}$
6: $P := P'$
7: **end if**
8: Válasszunk P-ből véletlenszerűen egy (u,v) élt
9: $T_1.v := u$, $T_2.v := v$
10: $V_T := \{u,v\}$
11: $E_T := \{(u,v)\}$
12: QI_{G_1}-t egészítsd ki (u,v)-vel
13: $E(G_w) := E(G_w) \backslash (u,v)$
14: **while** $V_T \neq V(G_w)$ **do**
15: $P := \{(u,v)|(u,v) \in E(G_w) \wedge u \in V_T \wedge v \notin V_T\}$
16: (u,v) :=Élkiválasztás(G_w, P)
17: $V_T := V_T \cup \{v\}$
18: $E_T := E_T \cup \{(u,v)\}$
19: QI_{G_1}-t egészítsük ki (u,v)-vel
20: $Q := \{(u,v)|(u,v) \in E(G_w) \wedge u \in V_T \wedge (u,v) \notin E_T\}$
21: **while** $Q \neq \emptyset$ **do**
22: Jelölje (u,v) az így kapott sorrendben az 1. élt
23: Egészítsük ki QI_{G_1}-et (u,v)-re vonatkozó élfeltétellel és a fokszámfeltétellel
24: $Q := Q \backslash (u,v)$
25: **end while**
26: **end while**
27: $T := (V_T, E_T)$
28: **return** T, QI_{G_1}



4.5. ábra. G_1-ből elkészített G_w gráf

Algoritmus 5. Élkiválasztás(G_w, P)

Input : P : élek egy halmaza

G_w: G_1-ből az átlagos belső tartó alapján elkészített súlyozott gráf

V_T: feszítőfa csúcshalmaza

Output: e: P-beli él

1: $P_1 := \{(u,v)|(u,v) \in P \wedge \forall (u',v') \in P(\, \phi(u,v) \leq \phi(u',v'))\}$

2: $P := P_1$

3: **if** $|P| > 1$ **then**

4: $P' := \{(u,v)|(u,v) \in P \wedge \forall (u',v') \in P(\, Ind_{G_1}(V_T \cup \{u,v\}) \leq Ind_{G_1}(V_T \cup \{u',v'\}))\}$, ahol $Ind_{G_1}(V_T \cup \{u,v\})$ jelöli a $V_T \cup \{u,v\}$ csúcsok által feszített részgráf éleinek a számát G_1-ben

5: $P := P'$

6: **end if**

7: **if** $|P| > 1$ **then**

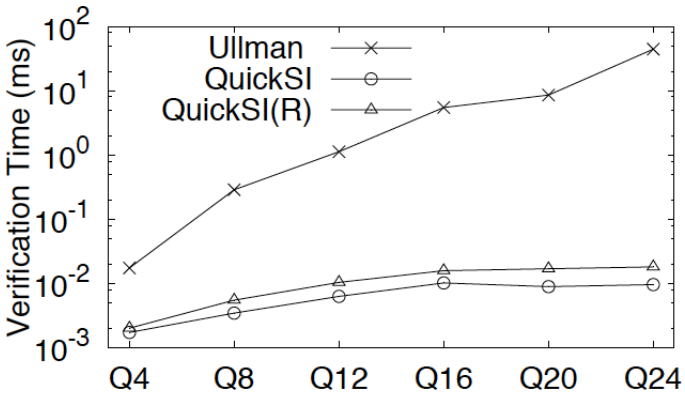
8: $P' = \{(u,v)|(u,v) \in P \wedge v \notin V_T \wedge \forall (u',v') \in P(v' \notin V_T \Rightarrow (deg_{G_1}(v) \leq deg_{G_1}(v')))\}$

9: $P := P'$

10: **end if**

11: Válasszunk P-ből véletlenszerűen egy (u,v) élt

12: **return** (u,v)



4.6. ábra. A különböző algoritmusok átlagos futásideje [20]

nálunk, 10.000 gráfot tartalmaz. Az adatbázis 62 különböző csúcscímkéjű gráfot tartalmaz, azonban a csúcsok címkéinek jelentős része C, N és O. A gráfoknak átlagosan 25.4 csúcsuk és 27.3 élük van. A módszereket különböző méretű gráfokon tesztelve vizsgáljuk a futási idő. Legyen Qi 1000 darab, i élű gráfot tartalmazó kisebb adatbázis. Jelölje QuickSI(R) a QuickSI-algoritmus futásidejét véletlenszerűen választott QI-sorozat esetén, QuickSI pedig a hatékony QI-sorozat esetén. Ekkor a különböző módszereket összehasonlítva a 4.6. ábrán szereplő eredményt kapjuk [20]. A futási időket összehasonlítva látjuk, hogy például $Q24$ esetén az Ullmann-algoritmus 5535-ször lassabb, mint a hatékony QI-sorozatot használó QuickSI-algoritmus. A 4.6. ábra alapján megállapítható, hogy a részgráf-izomorfia-vizsgálat költsége körülbelül feleakkora a QuickSI-módszer esetén, ha a véletlenszerű QI-sorozat helyett a hatékony QI-sorozattal dolgozunk.

5. fejezet

Szűrés az adatbázison

A továbbiakban megvizsgáljuk, hogy milyen módon határozhatjuk meg hatékonyan azon adatbázisbeli gráfokat, melyek részgráfként tartalmaznak egy adott gráfot. Az adatbázison előszűrést végzünk, mely során kiszűrjük azon gráfokat, melyeknek biztosan nem részgráfja az adott gráf. A szükséges definíciók ismertetése után egy általános előszűrő eljárást vizsgálunk H. Shang, Y. Zhang, X. Lin és J. Xu Yu cikke [20] cikke nyomán. Ehhez kisebb méretű gráfokat, ún. jellemző részgráfokat használunk. Ezen eljárás során részgráf-izomorfiát kell vizsgálni, ami – mint már korábban vizsgáltuk – költséges, és további kérdés, hogy mely jellemző részgráfokat alkalmazzunk az eljárás során. A jellemző részgráfok kiválasztásának különböző módszereit az egyes alfejezetekben tekintjük át. Megvizsgáljuk az R. Giugno és Dennis Shasha által publikált út alapú jellemző részgráfokat kiválasztó eljárást [11], az S. Zhang és M. Hu és J. Yang által javasolt fa alakú jellemző részgráfok kiválasztását [23], valamint a P. Zhao, J. Xu Yu és P. Yu cikkében ismertetett eljárást [24].

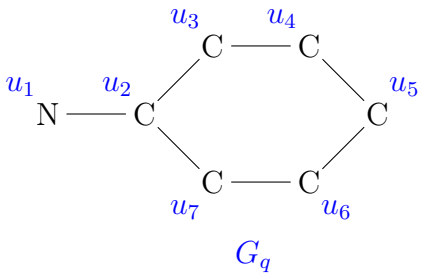
5.1. Általános előszűrő eljárás

5.1. Definíció *Részgráf-izomorfia-probléma adatbázisban: adott egy G_q query gráf és egy $D = \{G_1, \ldots, G_n\}$ gráfokat tartalmazó adatbázis. Határozzuk meg azon $G_i \in D$ gráfokat, melyekre $G_q \subseteq G_i$.*

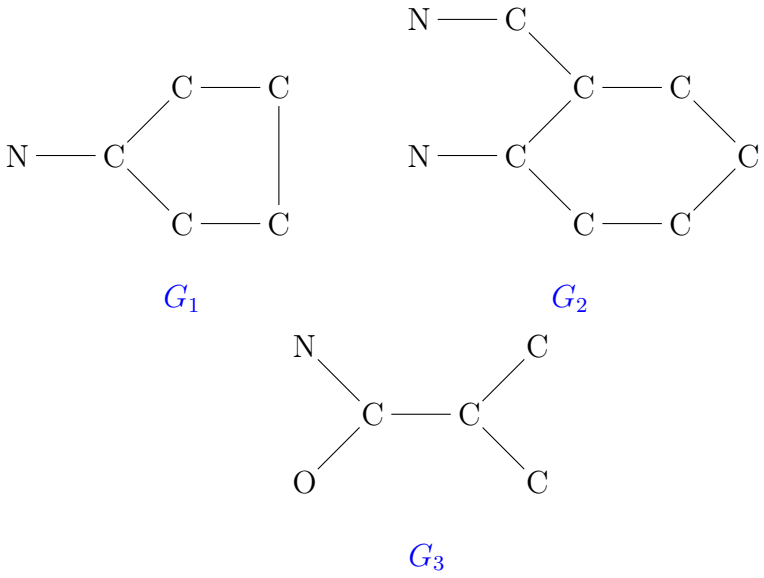
5.1. Példa

Legyen G_q az 5.1. ábrán szereplő gráf, a D adatbázis pedig álljon az 5.2. ábrán látható gráfokból. Ekkor $G_q \subseteq G_2$ és $G_q \nsubseteq G_1, G_3$.

Az adatbázis előszűrése során fontos szerepet játszanak az ún. jellemző részgráfok, melyek valamely adatbázisbeli gráfnak a részgráfjai.



5.1. ábra. Query gráf



5.2. ábra. Gráfadatbázis

Fontos kérdés, hogy egy adott adatbázis esetén hogyan válasszuk meg jól a jellemző részgráfokat. Ezen heurisztikákat a következő alfejezetekben tárgyaljuk. Minden egyes f_i jellemző részgráf esetén határozzuk meg a hozzá tartozó D_{f_i} halmaz, melyet a következőképp definiálunk: $D_{f_i} := \{G_j | G_j \in D \wedge f_i \subseteq G_j\}$, azaz D_{f_i} azon adatbázisbeli gráfokat tartalmazza, melyek részgráfként tartalmazzák f_i-t.

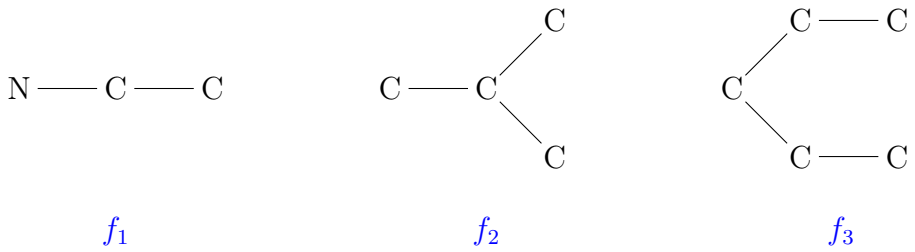
5.2. Példa

Az 5.1 példa esetén legyenek a jellemző részgráfok a 5.3. ábrán láthatóak. A fenti példa esetén a következő halmazokat kapjuk:

$D_{f_1} = \{G_1, G_2, G_3\}$

$D_{f_2} = \{G_2, G_3\}$

$D_{f_3} = \{G_1, G_3\}$



5.3. ábra. A jellemző részgráfok

Habár a fenti halmazok kiszámítása költséges – részgráf-izomorfiát kell vizsgálnunk – , de adatbázisban tárolva nem kell minden egyes részgráf-izomorfia vizsgálat esetén újra meghatározni ezen halmazokat. Fontosnak tartjuk hangsúlyozni, hogy ezen halmazok kiszámítása előfeldolgozási időben történik, nem keresési időben. Számítsuk ki $D_q := D_{G_q}$-t is. Ekkor elegendő azon gráfok esetén elvégezni a részgráf-izomorfia-tesztet, melyek tartalmazzák azon kisméretű, gyakori gráfokat részgráfként, melyeket G_q is. A jelölt halmazt a későbbiekben C_q-val jelöljünk. Megjegyezzük, hogy molekulagráfok esetén további egyszerű szűrési feltételek alapján adódnak (pl.: $C, N, O, \ldots$ atomok száma, kötések száma típusonként).

5.3. Példa

A fenti példa esetén $D_q = \{f_1, f_3\}$ *és* $C_q = D_{f_1} \cap D_{f_2} = \{G_1, G_2, G_3\} \cap \{G_2, G_3\} = \{G_1, G_2\}$.

A fent elmagyarázott eljárás pszeudokódját írja le a 6.. algoritmus, melynek során azon G gráfokat tartalmazó halmazt, melyre $G \in D$ és $G_q \subseteq G$, R-rel jelöltük.

A későbbiekben a korábban felvetett problémának, a jellemző részgráfok kiválasztásának néhány módszerét példákkal szemléltetve mutatjuk be. A jellemző részgráfok kiválasztását indexelésnek nevezzük. A jellemző részgráfokkal történő előszűrést az motiválja, hogy a query magas feldolgozási költségét egy egyszeri indexkészítési fázisba toljuk el. Adott G_q query gráf és $D = \{G_1, \ldots, G_n\}$ adatbázis esetén a részgráf-izomorfizmus probléma C összköltsége a jellemző részgráfokkal, $F = \{f_1, \ldots, f_m\}$-mel történő szűrés esetén $C = C_f + |C_q| \cdot C_v$, ahol C_f a szűrési költség a $C_q = \bigcap_{f_i \in F \wedge f_i \subseteq G_q} D_{f_i}$ egyenlőségen alapulva, és C_v jelöli a részgráf-izomorfia-vizsgálat átlagos költségét G_q és $G_j \in D$ esetén.

Legyen D egy rögzített adatbázis. Célunk $|C_q|$ minimalizálása. Látszólag C_q akkor lesz minimális, ha az összes, adatbázisban előforduló részgráfot kiválasztjuk. Ekkor azonban $|F|$ túlságosan nagy lehet, ezáltal F memóriában való tárolását ne-

Algoritmus 6. Szűrés(Q, I, D)

Input : G_q : query gráf

I : jellemző részgráfokat tartalmazó halmaz

D : gráfadatbázis

Output: R: azon G gráfokat tartalmazó halmaz, melyre $G \in D$ és $G_q \subseteq G$

1: $D_q := \{f_i | f_i \in I \wedge f_i \subseteq G_q\}$
2: $C_q := \bigcap_{f_i \in D_q} D_{f_i}$
3: $R := \emptyset$
4: **for each** $G \in C_q$ **do**
5: **if** $G_q \subseteq G$ **then**
6: $R := R \cup \{G\}$
7: **end if**
8: **end for each**
9: **return** R

hezíti, és a szűrési költséget, C_f-et is megnöveli. Másrészről, $|F|$ csökkentése C_f csökkenéséhez, de C_q növekedéséhez vezet. Mielőtt ismertetnénk elvárásainkat egy jó minőségű indexeléssel szemben, bevezetjük egy gráf nyesési erejének fogalmát adott D adatbázis esetén.

5.2. Definíció *Legyen $f_i \in F$. f_i nyesési erején az*

$$E_{ny}(f_i) = \frac{|D| - |\{G_j | G_j \in D \wedge f_i \subseteq G_j\}|}{|D|}$$

számot értjük.

Megjegyzés. *A definíció alapján $0 \leq E_{ny}(f_i) \leq 1$. Mivel f_i-t legalább egy $G_j \in D$ gráf tartalmazza részgráfként, így $E_{ny}(f_i) = 1$ nem lehetséges. Abban az esetben, ha $E_{ny}(f_i) = 0$, az f_i-vel történő szűrés nem csökkenti a C_q halmazt, mivel f_i minden adatbázisbeli gráfban megtalálható. Abban az esetben, ha $E_{ny}(f_i) \approx 1$, f_i-vel történő szűrés jelentősen csökkenti a C_q halmazt, amennyiben $f_i \subseteq G_q$ teljesül.*

Elvárásaink egy jó minőségű indexeléssel szemben:

1. Minden jellemző részgráf közepes nyesési erejű legyen. A túl nagy nyesési erejű részgráfok valószínűleg nem lesznek benne a query gráfban. Olyan jellemző részgráfok kellenek, amelyeknek elég nagy a nyesési erejük, és nagy valószínűséggel megtalálhatóak egy tipikus query gráfban.

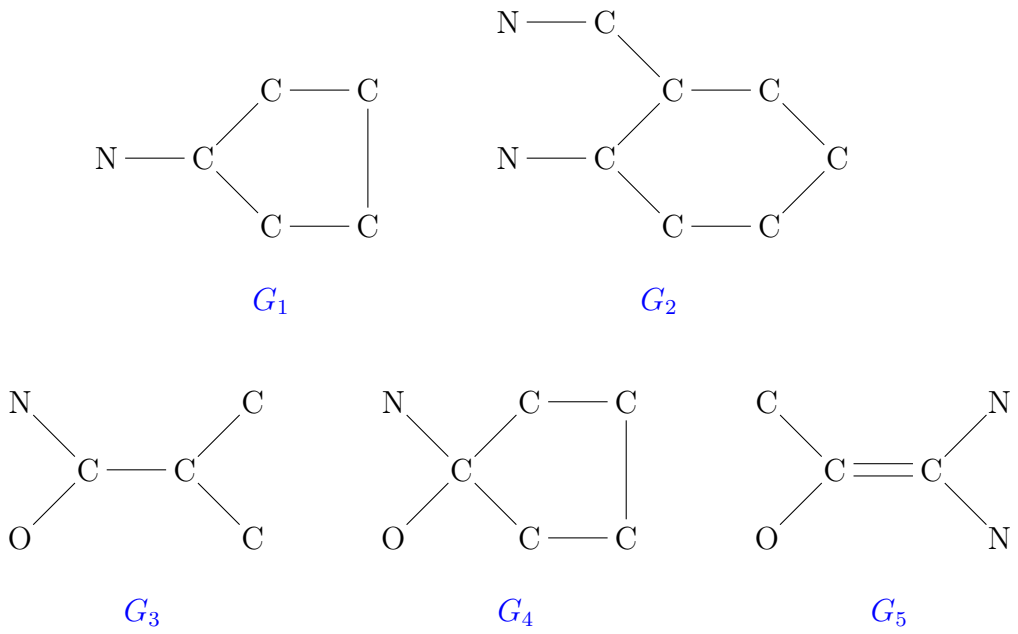
2. Lehetőség szerint kevés legyen belőlük, hogy a memóriában tömören tárolhatóak legyenek.

3. Az egyszeri számítási költség sem lehet akármekkora.

Fontosnak tartjuk megjegyezni, hogy jelenleg nincs univerzálisan legjobb módszer a jellemző részgráfok kiválasztására. Mindig az adott gráfok típusa és az adatbázis határozza meg, hogy mely módszer(ek) a hatékonyabb(ak) az adott esetben. A továbbiakban ezen módszerek közül mutatunk be néhányat.

5.4. Példa

A következő módszerek során legyen a gráfadatbázisunk az 5.4. ábrán látható.



5.4. ábra. Gráfadatbázis

5.2. Út alapú jellemző részgráfok

Ezen módszer esetén utak a jellemző részgráfok. Rögzítsünk egy kis, maximum 10 értékű l_p konstanst, s határozzuk meg az összes adatbázisbeli csúcsra a csúcsnál kezdődő, legfeljebb l_p hosszúságú utakat. Jelölje $F = \{f_1, \ldots, f_m\}$ az így kapott utakat tartalmazó halmazt. Az így kapott adatok alapján az 5.2 ábrán látható adatbázist építjük fel. Az f_i sorhoz tartozó G_j érték legyen az a szám, ahányszor f_i előfordul G_j-ben. A jellemző részgráfok szerkezetükből adódóan könnyen kutathatóak, kevés számúak, az adatbázisbeli értékek gyorsan kiszámolhatóak, azonban előfordulhat, hogy kevésbé jó nyesési erejűek. Tekintsük erre a következő példát.

Jellemző részgráf	G_1	G_2	G_3	G_4	G_5
$N-C$	1	2	1	1	2
$C-C$	5	7	3	5	1
$C-O$	0	0	1	1	1
$C=C$	0	0	0	0	1
$N-C-C$	2	3	1	2	0
$C-C-C$	5	8	3	5	0
$N-C-O$	0	0	1	1	0
$O-C-C$	0	0	1	2	1
$O-C=C$	0	0	0	0	1
$C-C=C$	0	0	0	0	1
$N-C-N$	0	0	0	0	1
$C=C-N$	0	0	0	0	2
$N-C-C-C$	2	5	2	2	0
$C-C-C-C$	5	8	0	5	0
$C-C-C-O$	0	0	2	2	0
$O-C=C-N$	0	0	0	0	2
$C-C=C-N$	0	0	0	0	2

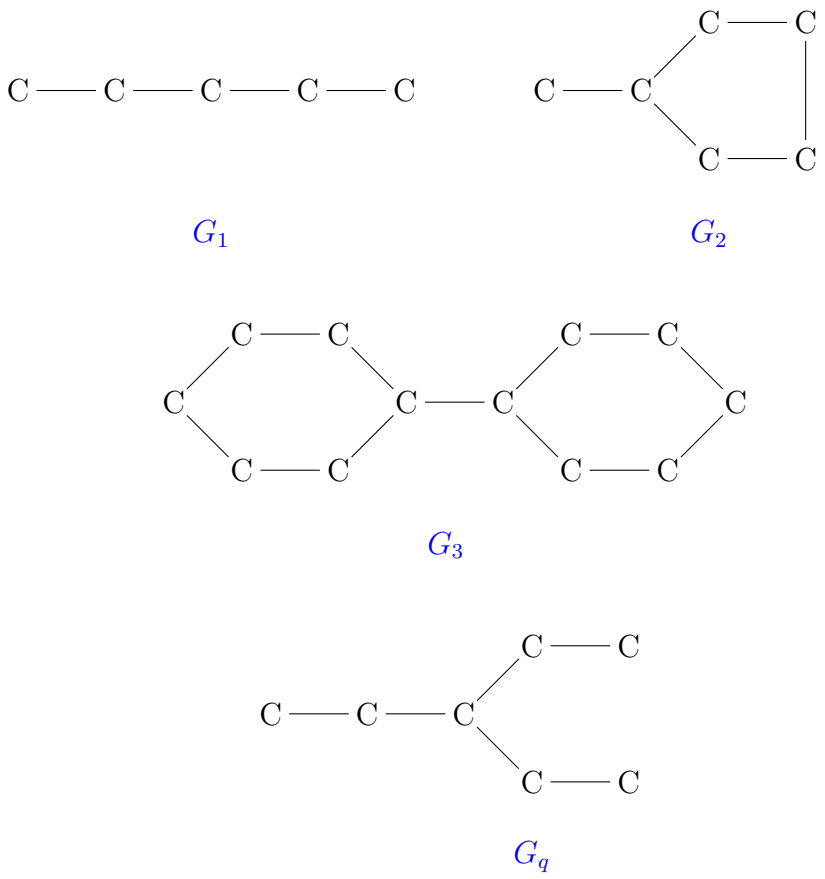
5.5. ábra. Tartalmazási táblázat

5.5. Példa

Tekintsük az 5.7. ábrát. Legyen $D := \{G_1, G_2, G_3\}$ *és* $l_p := 4$. *Ekkor* $C_q = D$, *de csak* $G_q \subseteq G_3$ *teljesül.*

Jellemző részgráf	G_1	$\ldots$	G_n
f_1			
f_2			
$\vdots$			
f_m			

5.6. ábra. Tartalmazási táblázat



5.7. ábra. Gráfadatbázis

5.3. Fa alapú indexelés

S. Zhang, M. Hu, és J. Yang cikkében [23] a fa alakú jellemző részgráfok kiválasztását vizsgálta. A jellemző részgráfok fa szerkezét az motiválja, hogy az utakhoz hasonlóan könnyen kutathatóak az adatbázisból, azonban több szerkezeti információt őriznek meg, mint az utak. Mivel egy adott adatbázis esetén a fák száma túlságosan nagy lehet, ezért csak a gyakori fákat választjuk ki az indexelés során. Ezután a fa centrumának vizsgálatával az általános szűrési eljárást, a 6. algoritmust kiegészítjük.

5.3. Definíció *Tetszőleges G gráf esetén $D_G := \{G_i | G_i \in D \wedge G \subseteq G_i\}$.*

5.4. Definíció *Egy G gráfot gyakorinak nevezünk, ha $|D_G| \geqslant \sigma$, ahol $\sigma > 0$ a felhasználó által megadott paraméter.*

A fő gondolatmenet a következő: válasszuk ki az adatbázis alapján a gyakori részfákat. Ezután a query gráfot bontsuk fel gyakori részfákra, majd a C_q halmazt ezen fák centrumaira vonatkozó egyenlőtlenség alapján tovább szűkíthetjük, melyet a későbbiek során részletesen ismertetünk.

Mivel a különböző fák száma n különböző címkéjű csúcs esetén n^{n-2} [22], és habár a nagy méretű fák nagy nyesési erejűek – kevés gráfban szerepelnek –, de egy átlagos gráfban nem fordulnak elő, ezért csak a gyakori részfákat választjuk ki az indexelés során, és ezért nem ugyanazt a σ értéket használjuk különböző csúcsszámú fák esetén.

Tapasztalat alapján definiáljuk σ-t a következőképpen:

$$\sigma(n) := \begin{cases} 1 & ha\ n \leq \alpha \\ 1 + \beta(n - \alpha) & ha\ \alpha < n \leq \gamma \\ \infty & ha\ n > \gamma \end{cases}$$

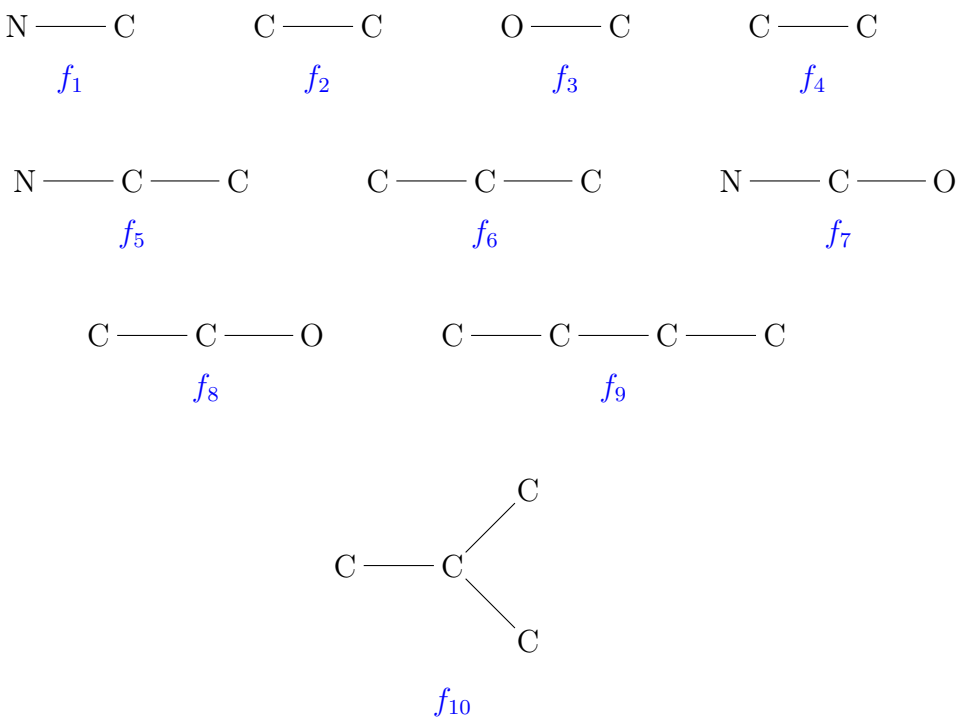
ahol n a fa csúcsainak száma, α, β, γ pedig pozitív paraméterek, melyeket az adatbázis alapján választunk meg. Az α, β, γ értékek megválasztására az alábbi heurisztikákat alkalmazhatjuk:

1. Legyen $\frac{s_q}{4} \leq \alpha \leq \frac{s_q}{2}$, ahol s_q jelöli a query gráf várható méretét.
2. Ha a legtöbb gyakori jellemző részgráf legfeljebb 10 csúcsú, akkor β legyen 1 vagy 2, egyéb esetben válasszuk β-t nagyobbnak, 5-nek vagy 6-nak.
3. $\gamma := min\{s_q, s_D\}$, ahol s_D-vel jelöljük az adatbázisbeli gráfok átlagos méretét.

Abban az esetben, ha az adatbázis nagy, a jellemző részgráfokat tartalmazó halmaz továbbra is nagy méretű lehet, így egy újabb lépésben töröljük ki azon fákat a jellemző részgráfokat tartalmazó halmazból, melyek részfáikhoz képest nem rendelkeznek extra nyesési erővel. Legyen a G_t fa valódi részfáinak halmaza: $\{t_1, \ldots, t_k\}$. Az 5.3. definícióból következik, hogy $|\bigcap_{i=1}^{k} D_{t_i}| \geq |D_{G_t}|$. Amennyiben $|\bigcap_{i=1}^{k} D_{t_i}| = |D_{G_t}|$, G_t-t törölhetjük a jellemző részgráfokat tartalmazó halmazból, mivel D_G kiszámítható $\{t_1, \ldots, t_k\}$ segítségével. Ezen megfigyelés alapján töröljük a kevésbé jelentős G_t fákat a a jellemző részgráfokat tartalmazó halmazból: $\frac{|\bigcap_{i=1}^{k} D_{t_i}|}{|D_{G_t}|} \leq \delta$, ahol $\delta > 1$ szűrési paraméter. Ha a jellemző részgráfokat tartalmazó halmaz túlságosan nagyméretű ahhoz, hogy a rendelkezésünkre álló memóriában tárolni tudjuk, α-t és γ-t fokozatosan csökkentve, δ-t növelve a halmaz mérete csökken.

5.6. Példa

Az 5.4. ábrán látható adatbázis esetén $s_D = 7$, $s_q = 5$. $\alpha = 2, \beta = 1, \gamma = 5$ választással az 5.8. ábrán látható jellemző részgráfokat kapjuk.



5.8. ábra. Kiválasztott jellemző részgráfok

Mivel a jellemző részgráfok fa szerkezetűek, és fa esetén a gráf centruma egyértelmű, a C_q halmazt tovább csökkenthetjük. Az eljárás ismertetéséhez szükségünk van néhány további fogalom definiálására.

5.5. Definíció *Legyen G egy gráf és $u \in V(G)$ csúcs. Jelöljük $d_{tav}(u)$-val az u-tól legtávolabb lévő csúcs távolságát, azaz $d_{tav}(u) := \max\{d(u,v)|v \in V(G)\}$, ahol $d(u,v)$ jelöli az u és a v csúcs távolságát.*

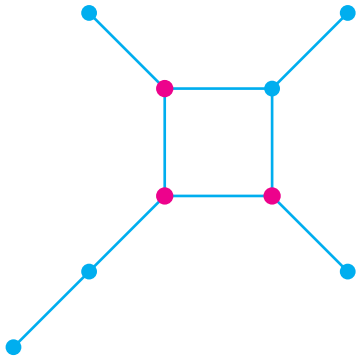
5.6. Definíció *Egy $G = (V, E)$ gráf centrumán, centrum(G)-n azon $u \in V$ csúcsokat tartalmazó halmazt értjük, melynek elemeire $d_{tav}(u)$ minimális, azaz*

$$centrum(G) := \{u|u \in V(G) \wedge \forall v \in V(G)\ d_{tav}(u) \leq d_{tav}(v)\}.$$

5.7. Példa

Tekintsük az 5.9 ábrán látható gráfot. A gráf centrumát bíborvörös színnel jelöltük.

5.7. Tétel *Egy G=(V,E) fa gráfnak a centruma egy csúcsot vagy két, szomszédos csúcsot tartalmazó halmaz.*



5.9. ábra. Példa gráf centrumára

Bizonyítás: A fa centrumát meghatározhatjuk az alábbi rekurzív eljárás segítségével, mely egyben a tételt is bizonyítja. Kezdetben legyen $T := G$. Minden lépésben töröljük az 1 fokszámú csúcsokat és a hozzá kapcsolódó élt a T fából. Ismételjük ezen eljárást addig, amíg csak egy vagy két, szomszédos csúcsot kapunk. Mivel T fa, ezért minden lépés során legalább 2 csúcsot törlünk, és minden törlés után továbbra is fát kapunk. Így az algoritmus véges sok lépés után leáll, meghatározva a G fa centrumát. □

5.8. Lemma *Tegyük fel, hogy $G_1 \subseteq G_2$, és G_1-et illetve G_2-t felbontottuk jellemző részfákra. Jelölje $\{t_1, t_2, \ldots, t_m\}$ azon jellemző részfákat tartalmazó halmazt, melyek G_1 és G_2 felbontásában is szerepelnek. Ekkor $d_{G_1}(centrum(t_i), centrum(t_j)) \geq d_{G_2}(centrum(t_i), centrum(t_j))$ $\forall 1 \leq i \neq j \leq m$.*

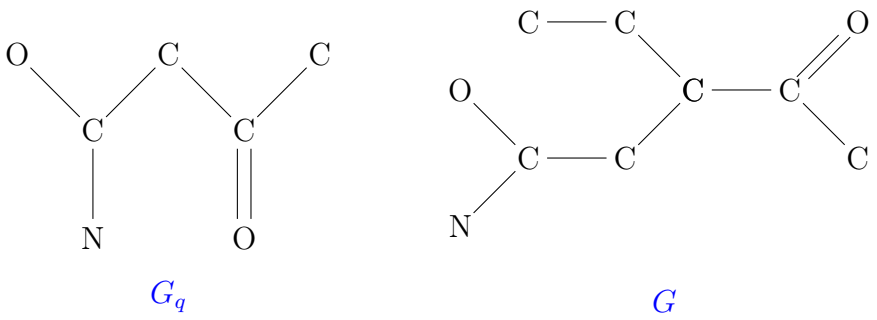
Bizonyítás: Mivel $G_1 \subseteq G_2$ a feltevés szerint, ezért G_2-nek van olyan G részgráfja, mely izomorf G_1-gyel. Ekkor bármely $u, v \in V(G_1)$ esetén u és v távolsága G_1-ben megegyezik u és v képének távolságával G-ben. Mivel G G_2-nek a részgráfja, így $d_G(u, v) \geq d_{G_2}(u, v)$. Válasszuk u-t és v-t t_i, illetve t_j $(i \neq j)$ centrumának. Ezen u, v választás mellett éppen a bizonyítandó állítást kapjuk. □

Nézzünk egy konkrét példát az 5.8. lemma alkalmazására.

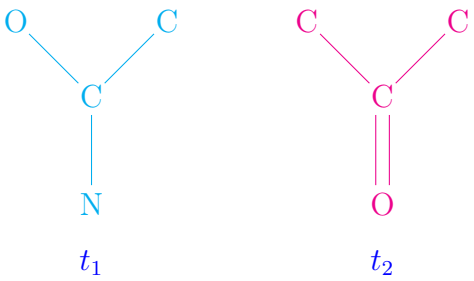
5.8. Példa

Legyen G_q és $G \in C_q$ az 5.10. ábrán látható gráfok. Bontsuk fel G_q-t t_1-re és t_2-re az 5.11. ábrán látható módon, majd határozzuk meg t_1 és t_2 centrumát, melyet az 5.12. ábrán jelöltünk. Ezután keressük meg t_1-et illetve t_2-t G-ben, melynek eredményét az 5.13. ábrán láthatjuk. Ellenőrizzük, hogy teljesül-e az 5.8. lemmában a centrumok távolságára vonatkozó egyenlőtlenség: $2 = d_{G_q}(centrum(t_1), centrum(t_2)) \not\geq$

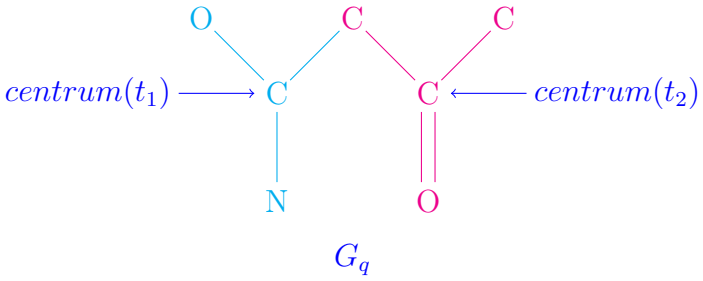
$d_G(centrum(t_1), centrum(t_2)) = 3$. *Ezen észrevétel alapján G-t törölhetjük a C_q halmazból.*



5.10. ábra.



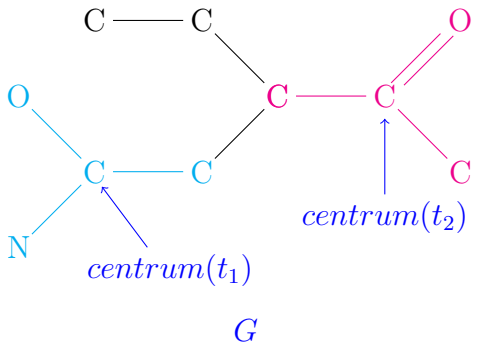
5.11. ábra.



5.12. ábra.

5.4. Tree+Delta

Jelen alfejezetben ismertetjük a P. Zhao, J. Xu Yu és P. S. Yu által kidolgozott indexelési eljárást [24]. A szerzők az AIDS Antiviral Screen Dataset [1] adatbázison



5.13. ábra.

vizsgálódva megfigyelték, hogy a gyakori jellemző részgráfok kb. 95%-a fa szerkezetű, így a nem fa jellemző részgráfok meghatározása során keletkező óriás költséget elkerülve egy új módszert dolgoztak ki a gyakori jellemző részgráfokat tartalmazó halmaz előállítására.

Az indexelés során először meghatározzuk a gyakori fa szerkezetű részgráfokat (melyek gyorsabban kutathatóak az adatbázisból, mint a nem fa szerkezetűek), majd újabb, költséges számítások nélkül ezen halmazt kiegészítjük néhány nagy nyesési erejű gráffal. Megjegyezzük, hogy jelen esetben ugyanazt a felhasználó által megadott értéket használjuk minden adatbázisbeli gráf esetén, ellentétben az előző, 5.3 alfejezetben ismertetett $\sigma(n)$-től. A σ értéke általában $\sigma = 0.1$.

Először határozzuk meg a gyakori jellemző részfákat rekurzióval. A legkisebb, 2 csúcsú fából kiindulva addig bővítjük a fát egy csúccsal és egy éllel, amíg az továbbra is gyakori (5.4. definíció, 33. oldal) marad. Egy közbülső lépés során, melynél f_i gyakori részfa, $C(f_i)$ jelöli azon f fákat tartalmazó halmazt, melyet úgy kapunk, hogy az f_i fát egy csúcs és egy él hozzávételével bővítünk, azaz

$$C(f_i) := \{f | f_i \subseteq f \wedge |V(f)| = |V(f_i)| + 1\}.$$

Mivel $f_i \subseteq f$ és $|D_{f_i}| \geq \sigma \cdot |D|$, így $|D_{f_i}| \geq |D_f|$. Ha a gráfoknak több csúcsuk lesz, mint a legnagyobb D-beli gráfnak, akkor már nem találjuk meg ezen gráfokat D-ben, ezért az algoritmus véges lépésben leáll. A fenti gondolatmenetet írja le a 7. és a 8. algoritmus.

A 8. algoritmus esetén két költséges algoritmus meghívása elkerülhetetlen:

1. Annak eldöntése, hogy f gyakori részgráf-e, azaz $|D_f| > \sigma \cdot |D|$, részgráf-izomorfia-vizsgálatot igényel a teljes adatbázison, melyről korábban megmutattuk, hogy NP-teljes feladat. A keresés során az alábbi heurisztikával jelentősen csökkenthetjük a keresés műveletigényét. Az egyre nagyobb f jellemző

Algoritmus 7. FFS(G,σ)

Input : D: gráfadatbázis

σ: a gyakori részgráf 5.4. definíciójában szereplő paraméter

Output: F: a gyakori jellemző részgráfokat tartalmazó halmaz

1: $F := \emptyset$
2: **for each** f gyakori részgráfra, melyre $|V(f)| = 2$ **do**
3: FFM(f, F, G, σ)
4: **end for each**
5: **return** F

Algoritmus 8. FFM(f, F, G, σ)

Input : f: gyakori jellemző részgráf

F: gyakori jellemző részgráfokat tartalmazó halmaz

G: gráfadatbázis

σ: a gyakori részgráf 5.4. definíciójában szereplő paraméter

Output: R is a set of matched graphs

1: **if** $f \in F$ **then**
2: **return**
3: **else**
4: $F := F \cup \{f\}$
5: $C(f) := \{f' | f \subseteq f' \wedge V(f')| = |V(f)| + 1\}$
6: **for each** $f' \in C(f)$ **do**
7: **if** $|D_{f'}| \geq \sigma \cdot |D|$ **then**
8: FFM(f', F, G, σ)
9: **end if**
10: **end for each**
11: **end if**

részgráfok meghatározása során eltároljuk a D_f halmazt, és ha $f' \subseteq f$, akkor $D_{f'} \subseteq D_f$. Így $D_{f'}$ meghatározásához elegendő a keresést csak a D_f-beli elemekre elvégzeni.

2. Az 1. sorban szükséges eldönteni, hogy f-et korábban kiválasztottuk-e F-be, mely gráfizomorfia-vizsgálatot igényel. Általános esetben nem tudni, hogy a gráfizomorfia probléma P-beli vagy NP-teljes, azonban fa esetén ezen vizsgálat műveletigénye $O(|V(f)|)$ [4].

Mivel G_q több gyakori jellemző részgráfot is tartalmazhat, így az 5.2. definícióhoz hasonlóan definiálhatjuk az $S = \{f_1, \ldots, f_m\}$ halmaz nyesési erejét.

5.9. Definíció *Legyen $S = \{f_1, \ldots, f_m\}$ gyakori jellemző részgráfokat tartalmazó halmaz. Az S halmaz nyesési erején a $E_{ny}(S) = \frac{|D| - |\bigcap_{i=1}^{m} D_{f_i}|}{|D|}$ számot értjük.*

5.10. Lemma *Legyen $f \in F$ gyakori jellemző részgráf, f gyakori jellemző részgráfjai pedig $S(f) = \{f_1, \ldots, f_m\}$. Ekkor $E_{ny}(f) \geq E_{ny}(S(f))$.*

Bizonyítás: Tudjuk, hogy

$$f \subseteq G_j \ \ \forall G_j \in D_f \tag{5.1}$$

valamint

$$f_i \subseteq f \ \ \forall f_i \in S(f) \tag{5.2}$$

(5.1) és (5.2) alapján

$$f_i \subseteq G_j, \ azaz \ G_j \in D_{f_i} \forall i \in [1, \ldots, m]. \tag{5.3}$$

Ebből következik, hogy

$$G_j \in \bigcap_{i=1}^{m} D_{f_i} \Rightarrow D_f \subseteq \bigcap_{i=1}^{m} D_{f_i} \Rightarrow |D_f| \leq |\bigcap_{i=1}^{m} D_{f_i}| \tag{5.4}$$

Az (5.4) egyenlőtlenséget alakítsuk át:

$$|D| - |D_f| \geq |D| - |\bigcap_{i=1}^{m} D_{f_i}| \tag{5.5}$$

Osszuk le mindkét oldalt $|D| > 0$-val:

$$\frac{|D| - |D_f|}{|D|} \geq \frac{|D| - |\bigcap_{i=1}^{m} D_{f_i}|}{|D|} \tag{5.6}$$

Az 5.9. definíció és a (5.6) alapján $E_{ny}(f) \geq E_{ny}(S(f))$. □

Habár a fa szerkezetű gyakori jellemző részgráfok közel azonos nyesési erővel rendelkeznek, mint a nem fa szerkezetűek, szükség lehet nem fa szerkezetű jellemző részgráfok kiválasztására is. Célunk ezen jellemző részgráfok meghatározása költséges adatbányászat nélkül. Legyen G egy nem fa szerkezű G_q-beli részgráf. Ha $E_{ny}(G) \approx E_{ny}(T(G_q))$, ahol $T(G_q)$ jelöli a G_q gráf gyakori jellemző részfáit, akkor G-t nem választjuk ki az indexelési eljárás során. Amennyiben $E_{ny}(G) \gg E_{ny}(T(G_q))$, szükségünk van G-vel való szűrésre az előszűrési eljárás során, mivel G jelentősen nagyobb nyesési erővel rendelkezik, mint $T(G_q)$.

Egy nem fa szerkezetű G gráf esetén vezessük be $\epsilon(G)$-t, mellyel G és G részfáinak nyesési erejét hasonlítjuk össze.

5.11. Definíció *Egy adott G gráf esetén a G gráf és $T(G)$ részfáinak nyesési erejének kapcsolatát írja le $\epsilon(G)$, melyet definiáljuk a következőképpen:*

$$\epsilon(G) := \begin{cases} \dfrac{E_{ny}(G) - E_{ny}(T(G))}{E_{ny}(G)} & ha\ E_{ny}(G) \neq 0 \\ 0 & ha\ E_{ny}(G) = 0 \end{cases}$$

Az 5.11. definíció alapján $0 \leq \epsilon(G) \leq 1$. Amennyiben $\epsilon(G) = 0$, akkor G és $T(G)$ nyesési ereje megegyezik. Abban az esetben, ha $\epsilon(G) = 1$, $T(G)$ nyesési ereje 0, ezért ha G nagy nyesési erejű, akkor mindenféleképpen ki kell választanunk az indexelési eljárás során.

5.12. Definíció *Egy G gráfot lényegesnek nevezünk, ha $\epsilon(G) \geq \epsilon_0$, ahol ϵ_0 a felhasználó által megadott $0 < \epsilon_0 < 1$ érték.*

A következőkben csak a lényeges, gyakori részgráfok kiválasztásával foglalkozunk. Vagyis a gyakoriságot limitáljuk mindkét irányból egy-egy konstanssal: csak olyan jellemző részgráfok kiválasztásával foglalkozunk, melyek ritkábbak, mint részgráfjai együttvéve, de ne legyenek túlságosan ritkák, azaz legyenek gyakoriak.

Adott G_q gráf esetén jelölje $D(G_q)$ azon G_i jellemző részgráfokat tartalmazó halmazt, melyek lényegesek. Az indexelés során nem választhatjuk ki minden egyes $G_i \in D(G_q)$ gráfot, mivel ez részgráf-izomorfizmus-vizsgálatot igényel a teljes adatbázison, ami – mint már korábban bemutattuk – költséges. Így amennyiben $G_i \in D(G_q)$ és $G_j \in D(G_q)$, valamint $G_i \subseteq G_j$, akkor G_i és G_j nyesési erejének összehasonlítása alapján döntjük el, hogy G_j-t beválasszuk-e a jellemző részgráfokat tartalmazó halmazba. Ha a két nyesési erő között nagy a különbség, G_j-t is beválasztjuk a jellemző részgráfokat tartalmazó halmazba, ellenkező esetben G_i nyesési ereje miatt nem szükséges G_j-t jellemző részgráfként kiválasztani. Mivel az egyes gráfok nyesési erejének meghatározásához részgráf-izomorfizmus-vizsgálatot kellene végezni az adatbázison, így a pontos értékek meghatározása helyett megelégszünk a nyesési erőre vonatkozó becslésekkel. A nyesési erőkre vonatkozó becslések meghatározásához szükségünk van néhány definícióra.

5.13. Definíció *Adott a G lényeges, gyakori gráf, melyre $G \in D(G_q)$. G előfordulási valószínűségén a $P(G) = \dfrac{|D_G|}{|D|} = \sigma_G$ számot értjük.*

G_j feltételes előfordulási valószínűsége a következőképpen definiálható.

5.14. Definíció *Adott két lényeges, gyakori gráf, G_i és G_j, melyekre $G_i \subseteq G_j$ és $G_i, G_j \in D(q)$. G_j feltételes előfordulási valószínűségén a*

$$P(G_j|G_i) = \frac{P(G_i \wedge G_j)}{P(G_i)} \overset{G_i \subseteq G_j}{=} \frac{P(G_j)}{P(G_i)} = \frac{|D_{G_j}|}{|D_{G_i}|}$$

számot értjük.

$P(G_i|G_j)$ értékének az 5.14. definíció alapján történő kiszámítása nem célravezető, mivel $|D_{G_i}|$ és $|D_{G_j}|$ pontos értékét jelenleg nem ismerjük. Ehelyett $T(G_i)$ és $T(G_j)$ felhasználásával becsüljük meg a keresett értéket.

5.15. Tétel *Adott két lényeges, gyakori gráf, G_i és G_j, melyekre $G_i \subseteq G_j$ és $G_i, G_j \in D(q)$. Ekkor*

$$P(G_j|G_i) \leq \frac{\sigma_{T(G_j)} - \epsilon_0}{\sigma \cdot (1 - \epsilon_0)} \tag{5.7}$$

és

$$P(G_j|G_i) \geq \frac{\sigma \cdot (1 - \epsilon_0)}{\sigma_{T(G_i)} - \epsilon_0} \tag{5.8}$$

Bizonyítás:

Mivel G_i gyakori jellemző részgráf, így

$$|D_{G_i}| \geq \sigma \cdot |D| \tag{5.9}$$

továbbá G_i lényeges gráf, így

$$\epsilon(G_i) \geq \epsilon_0 \tag{5.10}$$

(5.10) és az 5.2. definíció alapján

$$\frac{E_{ny}(G_i) - E_{ny}(T(G_i))}{E_{ny}(G_i)} \geq \epsilon_0 \tag{5.11}$$

Ezen egyenlőtlenséget az 5.2. definíció alapján átírva kapjuk:

$$\frac{\frac{|D| - |D_{G_i}|}{|D|} - \frac{|D| - |D_{T(G_i)}|}{|D|}}{\frac{|D| - |D_{G_i}|}{|D|}} \geq \epsilon_0 \tag{5.12}$$

$$|D_{T(G_i)}| - |D_{G_i}| \geq \epsilon_0 \cdot |D| - \epsilon_0 \cdot |D_{G_i}| \tag{5.13}$$

$$|D_{T(G_i)}| - \epsilon_0 \cdot |D| \geq (1 - \epsilon_0)|D_{G_i}| \tag{5.14}$$

$$\frac{|D_{T(G_i)}| - \epsilon_0 \cdot |D|}{1 - \epsilon_0} \geq |D_{G_i}| \tag{5.15}$$

Hasonlóan kapjuk $|D_{G_j}|$-re

$$|D_{G_j}| \geq \sigma \cdot |D| \tag{5.16}$$

és

$$\frac{|D_{T(G_j)}| - \epsilon_0 \cdot |D|}{1 - \epsilon_0} \geq |D_{G_j}| \tag{5.17}$$

Az 5.14. definíció, (5.9) és (5.16) alapján

$$P(G_j|G_i) = \frac{|D_{G_j}|}{|D_{G_i}|} \leq \frac{\frac{|D_{T(G_j)}| - \epsilon_0 \cdot |D|}{1 - \epsilon_0}}{\sigma \cdot |D|} = \frac{\frac{|D_{T(G_j)}|}{|D|} - \epsilon_0}{\sigma \cdot (1 - \epsilon_0)} = \frac{\sigma_{T(G_j)} - \epsilon_0}{\sigma \cdot (1 - \epsilon_0)} \tag{5.18}$$

Hasonló gondolatmenettel kapjuk, hogy

$$P(G_j|G_i) = \frac{|D_{G_j}|}{|D_{G_i}|} \geq \frac{\sigma \cdot |D|}{\frac{|D_{T(G_i)}| - \epsilon_0 \cdot |D|}{1 - \epsilon_0}} = \frac{\sigma \cdot (1 - \epsilon_0)}{\frac{|D_{T(G_i)}|}{|D|} - \epsilon_0} = \frac{\sigma \cdot (1 - \epsilon_0)}{\sigma_{T(G_i)} - \epsilon_0} \tag{5.19}$$

□

Mivel $P(G_j|G_i)$ valószínűség, így $0 \leq P(.) \leq 1$. Ezen egyenlőtlenség és 5.15. tétel alapján kapjuk, hogy

$$\sigma_{T(G_i)} \geq \max\{\epsilon_0, \sigma + (1 - \sigma)\epsilon_0\}$$

és

$$\max\{\sigma, \epsilon_0\} \leq \sigma_{T(G_j)} \leq \sigma + (1 - \sigma)\epsilon_0$$

$P(G_j|G_i)$ pontos értékének költséges kiszámítása helyett megelégszünk az 5.15. tételben kapott becsléssel. Mivel a G_q gráf minden fa részgráfját ismerjük, így a becslésben szereplő $T(G_i)$ és a $T(G_j)$ halmazok hatékonyan meghatározhatóak.

Irodalomjegyzék

[1] Aids antiviral screen dataset. Website. `http://dtp.nci.nih.gov/docs/aids/aids_data.html`.

[2] Graph center. Website. `http://en.wikipedia.org/wiki/Graph_center`.

[3] Subgraph isomorphism problem. Website. `http://en.wikipedia.org/wiki/Subgraph_isomorphism_problem`.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1974.

[5] Villányi Attila. *Kémia a kétszintű érettségire*. Kemavill Bt., 2009.

[6] Horst Bunke, Pasquale Foggia, C. Guidobaldi, Carlo Sansone, and Mario Vento. A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 123–132. Springer-Verlag, 2002.

[7] C. J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11:13–21, 1982.

[8] 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, 2007.

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

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

[11] Rosalba Giugno and Dennis Shasha. Graphgrep: A Fast and Universal Method for Querying Graphs. 2002. Proceeding of the International Conference in Pattern recognition.

[12] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, STOC '74, pages 172–184. ACM, 1974.

[13] Fekete István. Gráfalgoritmusok. Website, 2004. `http://people.inf.elte.hu/fekete/docs_2/grafalg/grafalg.pdf`.

[14] Evgeny B. Krissinel and Kim Henrick. Common subgraph isomorphism detection by backtracking search. *Software–Practice & Experience*, 34:591–607, May 2004.

[15] Tichler Krisztián. Részgráf-izomorfizmus és kapcsolódó problémák a kémiai informatikában, 2011. Formális eszközök az informatikában szeminárium.

[16] José Luis López-Presa and Antonio Fernández Anta. Fast Algorithm for Graph Isomorphism Testing. In *SEA*, pages 221–232, 2009.

[17] George S. Lueker and Kellogg S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. *Journal of ACM*, 26:183–195, April 1979.

[18] E. M. Luks. Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time. *Journal of Computer and System Sciences*, 1982.

[19] Brendan D. McKay. Nauty, 1981. `http://cs.anu.edu.au/~bdm/nauty`.

[20] 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.

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

[22] Katona Gyula Y., Recski András, and Szabó Csaba. *A számítástudomány alapjai*. Typotex kiadó, 2006.

[23] Shijie Zhang, Meng Hu, and Jiong Yang. TreePi: A Novel Graph Indexing Method. 2011. Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference.

[24] Peixiang Zhao, Jeffrey Xu Yu, and Philip S. Yu. Graph Indexing: Tree + Delta >= Graph. 2007. VLDB '07, September 23-28, 2007, Vienna, Austria.