Algoritmusok a minimális lefogó ponthalmaz problémára

Szakdolgozat

Hegedűs Tímea
Alkalmazott matematikus szak

Témavezető:
Jordán Tibor, egyetemi docens

Eötvös Loránd Tudományegyetem
Természettudományi Kar
Operációkutatási Tanszék
Budapest, 2004

Tartalomjegyzék

1. Bevezetés

Ismert tény, hogy a minimális lefogó ponthalmaz probléma NP-teljes. Tehát nem számíthatunk arra, hogy polinomiális időben egzakt algoritmust kapjunk, hacsak nem P=NP. Néhány speciális esetben létezik polinomiális egzakt algoritmus, például páros gráfok esetén a minimális lefogó ponthalmaz mérete megegyezik a maximális párosítás méretével, és erre ismert polinomiális algoritmus, azonban most az általános esettel foglalkozunk.

Sokan foglalkoztak vele, hogy minél jobb algoritmusokat találjanak erre a feladatra. A kutatások egyik iránya az, hogy minél gyorsabb egzakt algoritmust keressenek, amely megmondja, hogy van-e k pontú lefogó ponthalmaz egy gráfban. A legelső ilyen algoritmus Busstól származik [3], ennek futási ideje $O(kn + 2^k k^{2k+2})$. Ezt aztán többen tovább fejlesztették. A legutóbbi ilyen eredmény $O(kn + 1,271^k k^2)$, Chen és Jia [4] algoritmusa, ők a problémát kisfokú gráfokra vezették vissza. Olyan gráfra, ahol minden pont foka legfeljebb 3, $O(1,2192^k k)$ idejű algoritmust találtak [5].

A másik irány az, hogy polinomiális idejű approximációs algoritmust találjanak. Ismert konstans faktor a 2, ezt a gráf paraméterei segítségével lehet javítani. Például ha χ a gráf kromatikus száma, akkor $2 - \frac{2}{\chi}$ faktort lehet elérni. Ha a maximális fok Δ, akkor $2 - \frac{2}{\Delta}$ az approximációs faktor. Síkgráfokra $\frac{3}{2}$-approximáció ismert. Bar-Yehuda és Even a [2]-ben egy $2 - \frac{\log\log n}{2\log n}$-approximációs algoritmust ismertetnek. Håstad [9] megmutatta, hogy $\frac{7}{6}$-nál jobb approximációs faktor nem érhető el.

Ezen szakdolgozat célja, hogy approximációs algoritmusokat mutassak a problémára, illetve annak súlyozott változatára, és ezeket az algoritmusokat összehasonlítsam. Egyes esetekben egy általánosabb feladatot oldok meg: vagy hipergráfban keresek lefogó ponthalmazt, vagy a halmazfedési problémával foglalkozom. Ezek a feladatok nagyon könnyen megfeleltethetők egymásnak, és az is könnyen látszik, hogy az algoritmusok hogyan néznek ki hagyományos gráfokon.

2. Az alapfeladat és kapcsolódó problémák

2.1. Definíciók, jelölések

Minimális lefogó ponthalmaz probléma gráfokon

Adott egy $G = (V, E)$ gráf. Legyen $n = |V|$, $m = |E|$. Jelölje $d(v)$ a $v \in V$ pont fokát.

Lefogó ponthalmazon olyan $C \subseteq V$ halmazt értünk, hogy $\forall(uv) \in E$ élre u és v legalább egyike C-ben van. Célunk olyan lefogó ponthalmazt keresni, amely minimális méretű.

Ha a csúcsokon adott egy $w : V \to \mathbb{R}_+$ súlyozás, akkor minimális összsúlyú lefogó ponthalmazt keresünk, vagyis olyat, amelyre $\sum\{w(v) : v \in C\}$ minimális.

Jelölje C^* a G egy optimális lefogó ponthalmazát, OPT pedig az optimum értékét. Azaz súlyozatlan esetben $OPT = |C^*|$, súlyozott esetben $OPT = \sum\{w(v) : v \in C^*\}$. Néhány esetben, ahol több gráf is szerepel, az egyértelműség kedvéért a G gráfon az optimum értékét $OPT(G)$ jelöli.

Minimális lefogó ponthalmaz probléma hipergráfokon

Adott egy $G = (V, E \subseteq 2^V)$ hipergráf. Legyen $n = |V|$, $m = |E|$, valamint f a legnagyobb hiperél mérete. (Természetesen hagyományos gráfokra $f = 2$.) Ezen kívül legyen M a G mérete, vagyis a hiperélek méreteinek összege.

Lefogó ponthalmazon olyan $C \subseteq V$ halmazt értünk, hogy minden $e = (v_1, \ldots, v_k) \in E$ élre a v_i-k legalább egyike C-ben van. Célunk olyan lefogó ponthalmazt keresni, amely minimális méretű.

Ha a csúcsokon adott egy $w : V \to \mathbb{R}_+$ súlyozás, akkor minimális összsúlyú lefogó ponthalmazt keresünk, vagyis olyat, amelyre $\sum\{w(v) : v \in C\}$ minimális. Az optimális halmazokat, illetve az optimum értékét ugyanúgy jelöljük, mint ahogy hagyományos gráfok esetén.

Minimális halmazfedési probléma

Adott $\mathcal{H} = (U, \mathcal{S})$, ahol U egy alaphalmaz, $\mathcal{S} = \{S_1, \ldots, S_n\}$ halmazrendszer U-n úgy, hogy $\bigcup S_j = U$. Legyen $m = |U|$, és jelölje N a legnagyobb halmaz elemszámát, valamint f legyen a legnagyobb szám, ahány halmazban egy elem előfordul. Adott továbbá egy $w : \mathcal{S} \to \mathbb{R}_+$ súlyfüggvény a halmazokon.

Halmazfedésen olyan $\mathcal{C} \subseteq \mathcal{S}$ halmazrendszert értünk, amely minden elemet lefed U-ból. Célunk minimális összsúlyú halmazfedést találni, vagyis olyat, amelyre $\sum\{w(S) : S \in \mathcal{C}\}$ minimális.

Ugyanez a feladat indexhalmazokkal megfogalmazva: Ha $I = \{1, \ldots, n\}$ az $\mathcal{S}$-beli halmazok indexeinek halmaza, akkor olyan $J \subseteq I$ indexhalmazt keresünk, amelyre $\sum\{w(S_j) : j \in J\}$ minimális.

Jelölje $\mathcal{C}^*$ a $\mathcal{H}$ egy optimális fedését, J^* ennek indexhalmazát, és OPT az optimum értékét.

2.2. A lefogó ponthalmaz probléma és a halmazfedési feladat kapcsolata

A halmazfedési feladat a lefogó ponthalmaz feladatnak egy általánosítása a következő értlemben: Ha adott a $G = (V, E)$ gráf (vagy $G = (V, E \subseteq 2^V)$ hipergráf) a w súlyfüggvénnyel a pontokon, akkor legyen $\mathcal{H} = (U, \mathcal{S})$ a következő. Feleltessük meg U-nak az élek halmazát, $\mathcal{S}$-nek a pontok halmazát. Egy S halmazban azok az elemek vannak, amelyeknek megfelelő élek az S-nek megfelelő v pontra illeszkednek. Egy S halmaz $w(S)$ súlya a neki megfelelő v pont $w(v)$ súlya legyen, elemszáma, $|S|$ pedig v foka, $d(v)$. Ha találtunk egy $\mathcal{C}$ halmazfedést a halmazrendszerben, akkor az ennek megfelelő C lefogó ponthalmaz a (hiper)gráfban.

Súlyozatlan eset

3. Maximális párosítás keresése

A legegyszerűbb algoritmus a minimális lefogó ponthalmaz problémára Gavriltól származik. Keressünk egy tartalmazásra nézve maximális párosítást, legyen ez M, és az M-beli élek végpontjait válasszuk be C-be. Ekkor C nyilván lefogó ponthamaz, mert ha lenne olyan él, amelynek egyik végpontja sincs C-ben, akkor ezt az élt hozzá tudnánk venni M-hez, ellentétben M maximalitásával. Mivel az M-beli élek lefogásához legalább $|M|$ darab pont kell, így egy optimális C^* lefogásra $|C^*| \geq |M|$ teljesül. Tudjuk, hogy $|C| = 2|M|$, ebből pedig $|C| \leq 2|C^*|$ következik. Vagyis Gavril algoritmusa 2-approximáció.

C. Savage [13] egy további megfigyelést tett erről az algoritmusról. Legyen M' egy maximális méretű párosítás. Erre is igaz, hogy $|C^*| \geq |M'|$. Ekkor a következő egyenlőtelnség-sorozat teljesül:

$$|M| \leq |M'| \leq |C^*| \leq |C| = 2|M|.$$

Ebből látszik Gavril eredménye is, vagyis $|C| \leq 2|C^*|$. Ennél többet állíthatunk:

$$\frac{|C|}{|C^*|} \cdot \frac{|C^*|}{|M'|} \cdot \frac{|M'|}{|M|} = 2.$$

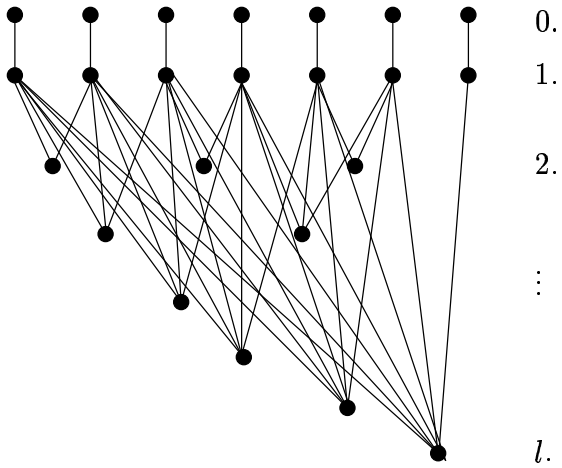
Ebből az látszik, hogy ha C relatívan rossz approximáció C^*-hoz, akkor $|C^*|/|M'|$ relatíven kicsi. A legegyszerűbb példa ennek illusztrálására a teljes páros gráf, $K_{n,n}$. Itt egy tartalmazásra nézve maximális párosítás egyben maximális méretű párostítás is, sőt teljes párosítás. Így $|C| = 2n$, azonban egy optimális lefogáshoz elég az egyik osztály pontjait vennünk, vagyis $|C^*| = n$.

4. Mohó algoritmus

Ez az algoritmus egy egyszerű ötleten alapul: minden lépésben egy olyan pontot válasszunk, amely a legtöbb élt lefogja. Vagyis mindig egy maximális fokú pontot keresünk. A kiválasztott pontot elhagyjuk a gráfból, és a következő lépésben a kisebb gráfban keresünk tovább. Bár ez egy nagyon ésszerűnek tűnő ötlet, sajnos nagyon rossz eredményt is tud adni.

4.1. Tétel. *A mohó algoritmus $O(\ln n)$ approximáció.*

A tételt most nem bizonyítom, mert a súlyozott esetben ezt általánosabban megteszem. Viszont szerepeljen itt egy példa, amely azt mutatja, hogy az $O(\ln n)$ faktor éles.



Amint látszik, a legnagyobb fokú pont az l-edik szinten van, a foka l. Ezt elhagyva a legnagyobb fokú pont az $l-1$-edik szinten van, foka $l-1$. És így tovább, ha az $l, l-1, \ldots, l-i$ $(0 \leq i \leq l-3)$ szintekről már kiválasztottuk a pontokat, akkor a legnagyobb fokú pont az $l-i-1$-edik szinten van, foka $l-i-1$. Ha pedig már a 2. szintről is kiválasztottuk a pontokat, akkor marad egy párosítás a 0. és 1. szint között, amit lefog az 1. szint. Vagyis összesen

$$L = \left\lfloor \frac{l}{l} \right\rfloor + \left\lfloor \frac{l}{l-1} \right\rfloor + \cdots + \left\lfloor \frac{l}{2} \right\rfloor + l = O(l \ln l)$$

pontot választottunk, holott az 1. szint önmagában is lefogó ponthalmaz, és l pontból áll.

5. DFS feszítőfák és lefogó ponthalmazok

5.1. Gyökeres fák

Carla Savage [13] a dfs feszítőfák és a lefogó ponthalmazok között mutatott meg összefüggést. Ebben a fejezetben ezt vizsgáljuk.

Vezessünk be néhány jelölést. Legyen $T = (V, E, r)$ egy r gyökerű fa. $v \in V$-re legyen $t(v)$ a v távolsága r-től. $(wv) \in E$ és $t(w) = t(v) + 1$ esetén v a w *szülője*, w pedig a v *gyereke*. Minden $w \neq r$ csúcsnak van szülője, ezt $f(w)$ jelöli. Két csúcs, w_1 és w_2 *testvérek*, ha $\exists v \in V : v = f(w_1) = f(w_2)$.

Legyen M egy párosítás T-ben. Azt mondjuk, hogy egy $v \in V$ csúcs M *által fedetlen*, ha v-re nem illeszkedik egyetlen M-beli él sem. Nevezzünk *tökéletes párosításnak* egy olyan M párosítást, amely kielégíti a következő tulajdonságot:
Ha T-nek egy $w \neq r$ csúcsa M által fedetlen, akkor w valamely u testvérére $(u, f(w)) \in M$.

Könnyen látható, hogy ha M tökéletes párosítás T-ben, akkor maximális párosítás.

5.1. Tétel. *Ha T egy dfs feszítőfa G-ben, $M(T)$ és $M(G)$ maximális párosítás T-ben illetve G-ben, akkor $|M(G)|/|M(T)| \leq 2$.*

5.2. Lefogó ponthalmazok és fák

Legyen M egy tökéletes párosítás T-ben, és legyen S a gyerekek, F a szülők halmaza M élein, azaz

$$S = \{w \in V | (w, f(w)) \in M\},$$
$$F = \{f(w) | w \in S\}.$$

Az S-ben és F-ben nem szereplő pontok M által fedetlenek.

5.2. Tétel. *F minimális lefogó ponthalmaz T-ben.*

Bizonyítás. Mivel $|F| = |M|$, ezért ha F lefogó ponthalmaz T-ben, szükségképpen minimális. Legyen (wv) a T egy éle úgy, hogy $f(w) = v$. Ha w fedetlen, akkor M tökéletessége miatt $v \in F$. Ha $w \in S$, akkor $v \in F$ az S és F definíciója szerint. Tehát vagy w, vagy v F-ben van. □

5.3. Tétel. *Legyen T dfs feszítőfa G-ben, $NL(T)$ a nem-levelek halmaza T-ben, és legyen $C^*(G)$ minimális lefogó ponthalmaz G-ben. Ekkor $NL(T)$ lefogó ponthalmaz G-ben, és $|NL(T)|/|C^*(G)| \leq 2$.*

Bizonyítás. Legyen $L(T)$ a levelek halmaza T-ben. Mivel T egy dfs feszítőfa, ezért G-nek nincs olyan éle, amelynek mindkét végpontja $L(T)$-ben van. Tehát $NL(T)$ lefogó ponthalmaz G-ben. T egy v csúcsára jelölje $g(v)$ a v gyerekeinek számát. Ekkor

$$|L(T)| = \sum_{v \in NL(T)} (g(v) - 1) + 1.$$

Legyen $M(T)$ egy tökéletes (és így maximális) párosítás T-ben, és legyen X a T-beli, $M(T)$ által fedetlen pontok halmaza. Egy $w \in X$, $w \neq r$ csúcs szülője, v egy nem-levél csúcs, amelyet $M(T)$ párosít w valamely testvérével. Így $g(v) \geq 2$. Mivel X-ben legfeljebb egy csúcs lehet T gyökere, az alábbit kapjuk:

$$|L(T)| \geq (|X| - 1) + 1 = |X|.$$

Mivel $|X| = n - 2|M(T)|$ és $|L(T)| = n - |NL(T)|$, ezért $|NL(T)| \leq 2|M(T)|$. Összegezve:

$$|M(T)| \leq |M(G)| \leq |C^*(G)| \leq |NL(T)| \leq 2|M(T)|, \qquad (1)$$

és így $|NL(T)|/|C^*(G)| \leq 2$. □

Figyeljük meg, hogy az (1) egyenlőtlenség-sorozat mutatja, hogy $|M(G)|/|M(T)| \leq 2$. Ezzel megkaptuk az 5.1 Tétel bizonyítását.

5.4. Következmény. *Ha $C^*(G)$ optimális lefogó ponthalmaz G-ben, T pedig a G egy dfs feszítőfája, melynek $C^*(T)$ optimális lefogó ponthalmaza, akkor $|C^*(G)|/|C^*(T)| \leq 2$.*

Bizonyítás. Legyen $M(T)$ egy maximális párosítás T-ben, $NL(T)$ pedig a nem-levelek halmaza. Az 5.3 Tétel bizonyításából

$$|C^*(T)| \leq |C^*(G)| \leq |NL(T)| \leq 2|M(T)| = 2|C^*(T)|.$$

□

A fenti korlát éles. Ennek bemutatására legyen $G = K_n$, ahol n páratlan. Ekkor $|C^*(G)| = n - 1$. Egy dfs feszítőfa egy $n - 1$ hosszú út, amelyre $|C^*(T)| = (n-1)/2$.

5.3. Néhány észrevétel

Az (1) egyenlőtlenség-sorozatból megkaptuk az $|M(G)|/|M(T)| \leq 2$, $|NL(T)|/|C^*(G)| \leq 2$, valamint $|C^*(G)|/|M(G)| \leq 2$ egyenlőtlenségeket. Ezeknél erősebb eredményt kapunk, ha $|NL(T)|/|M(T)| \leq 2$-t is figyelembe vesszük:

$$\frac{|NL(T)|}{|C^*(G)|} \cdot \frac{|C^*(G)|}{|M(G)|} \cdot \frac{|M(G)|}{|M(T)|} \leq 2.$$

Ebből az látszik például, hogy ha $|NL(T)|$ relatívan rossz approximáció $|C^*(G)|$-hez, akkor $|M(T)|$ relatívan jó approximáció $|M(G)|$-hez. Ezen kívül $|C^*(G)|/|M(G)|$ is kicsi kell legyen, ez pedig már független a T választásától.

Figyeljük meg, hogy a tétel bizonyításában r nem-levél T-ben, még akkor sem, ha elsőfokú pont. Ennek fontosságának igazolásához a legegyszerűbb példa, ha G egy háromszög. Ekkor T egy 2 élből álló út, a gyökér is elsőfokú, de az egyetlen másodfokú pont még nem alkot lefogást. Általánosabban akkor lehet probléma, ha r-nek G-ben van másodfokú u szomszédja. Ekkor ha egy (rv) élen indul a dfs fa építése, és r összes szomszédja elérhető v-ből a fában, akkor r elsőfokú lesz. Mivel u is elsőfokú, tehát levél, így az (ru) él lefogásához r-et nem-levélnek kell tekintenünk.

Természetesen előfordul olyan példa is, amikor r elsőfokú, és tekinthető levélnek. Ilyen egy csillag, ahol r az egyik ág vége.

6. Páratlan körök és lefogó ponthalmazok

6.1. Páratlan körök

Ebben a fejzetben Nagamochi és Ibaraki [12] algoritmusát vizsgáljuk.

Jelölje egy $G = (V, E)$ gráf H részgráfjának ponthalmazát $V(H)$, élhalmazát $E(H)$. Egy K kört *páratlan körnek* nevezünk, ha $|V(K)|$ páratlan, különben pedig *páros körnek*. Nyilván egy K páratlan körben az optimális lefogó ponthalmaz elemszáma, $OPT(K) = (|V(K)|+1)/2$. Mivel páros gráfban egy optimális lefogó ponthalmaz $O(n^{1/2}m)$ időben megtalálható, ezért most olyan gráfokat vizsgálunk, melyekben van páratlan kör.

6.1. Definíció. *A G gráf egy K köréhez egy e élt* húrnak *nevezünk, ha e mindkét végpontja $V(K)$-ban van, de $e \notin E(K)$. A K kört* húrmentesnek *nevezzük, ha nincs húrja.*

6.2. Lemma. *Legyen K a legrövidebb páratlan kör $G = (V, E)$-ben. Ekkor K húrmentes és*

$$\forall v \in V - V(K)\text{-nak legfeljebb három szomszédja van } V(K)\text{-ban.} \quad (2)$$

Bizonyítás. K nyilván húrmentes, mert különben tartalmazna egy rövidebb páratlan kört. Legyen $k = |K|$. $k = 3$-ra a lemma állítása igaz. Tehát legyen most $k \geq 5$, megmutatjuk, hogy minden $v \in V - V(K)$ csúcsnak legfeljebb két szomszédja van $V(K)$-ban. Tegyük fel, hogy a $v \in V - V(K)$ csúcsnak van két szomszédja $V(K)$-ban, u_1 és u_2. Ekkor $(u_1u_2) \notin E(K)$, mert különben $\{v, u_1, u_2\}$ egy háromszög lenne, ellentmondásban $k \geq 5$-tel, ami a legrövidebb páratlan kör hossza. Továbbá kell, hogy legyen egy $w \in V(K) - \{u_1, u_2\}$ csúcs, amelyre $(u_1w), (wu_2) \in E(K)$, mert különben lenne egy K-nál rövidebb páratlan kör $E(K) \cup \{(u_1w), (wu_2)\}$-ben.

Tegyük fel, hogy van egy $v' \in V - V(K)$ csúcs, amelynek három szomszédja van $V(K)$-ban, u_1, u_2 és u_3. A fenti megfigyelés alapján $\{u_1, u_2, u_3\}$ nem szomszédosak egymással, és bármely kettő $u_i, u_j \in \{u_1, u_2, u_3\}$-hoz van egy $w_{ij} \in V(K) - \{u_1, u_2, u_3\}$ csúcs, amelyre $(u_iw_{ij}), (w_{ij}u_j) \in E(K)$. Ezek

a w_{ij}-k különbözőek és nem szomszédosak. Ebből következik, hogy $|K| = 6$, ellentmondásban azzal, hogy K páratlan. □

6.3. Definíció. *Nevezzünk egy páratlan kört* szabályosnak, *ha teljesíti a (2) tulajdonságot.*

6.2. Az algoritmus szabályos körök segítségével

Az algoritmus alapötlete az, hogy mivel egy darab páratlan körben, illetve páros gráfban könnyen tudunk találni lefogó ponthalmazt, ezért páratlan köröket keresünk.

6.1. Algoritmus.

$C := \emptyset$, $G_0 := G$;
while(a) és (b)nem teljesül G_0-ban:
(a) G_0 egy páratlan körből és néhány izolált pontból áll
(b) G_0 páros gráf **do**
$K :=$ egy szabályos páratlan kör G_0-ban;
$C := C \cup V(K)$;
$G_0 := G_0 - V(K)$;
end
$C' :=$ minimális lefogó ponthalmaz G_0-ban;
$C := C \cup C'$;
Output: C.

6.4. Lemma. *$H = (V_1, V_2, E)$ páros gráfban $OPT(H) \geq |E(H)|/|V(H)|$.*

Bizonyítás. Legyen $m = |E|$, $p = |V_1|$ és $q = |V_2|$. Feltehető, hogy $p \leq q$. Tekintsünk egy $\pi : V_1 \to V_2$ leképezést, amelyre $u, v \in V_1$, $u \neq v$ esetén $\pi(u) \neq \pi(v)$. Összesen $\binom{q}{p}p!$ ilyen π van. Egy $(uv) \in E$ élre, melyre $u \in V_1$ és $v \in V_2$, $\binom{q-1}{p-1}(p-1)!$ olyan leképezés van, ahol $\pi(u) = v$. Így a skatulya-elv szerint van egy π leképezés, amelyre $\{(u, \pi(u)) | u \in V_1\}$ legalább $\left(m\binom{q-1}{p-1}(p-1)!\right) / \left(\binom{q}{p}p!\right) = m/q$ élt tartalmaz E-ben. Így a maximális párosítás mérete (amely páros gráf esetén megegyezik a minimális lefogó ponthalmaz méretével) legalább $m/q > |E(H)|/|V(H)|$. □

Legyen C egy lefogás $G = (V, E)$-ben. Ekkor egy tetszőleges $V' \subseteq V$ részhalmazra $|C \cap V'| \geq OPT(G[V'])$, hiszen $C \cap V'$ egy lefogás $G[V']$-ben. Így kapjuk a következő megfigyelést:

6.5. Lemma. *V-nek egy $\{V_1, \ldots V_r\}$ partíciójára teljesül a következő:*

$$OPT(G) \geq OPT(G[V_1]) + \cdots + OPT(G[V_r]).$$

Most vizsgáljuk meg, hogy az algoritmus milyen eredményt ad.

6.6. Tétel. *Egy $G = (V, E)$ gráfban az algoritmus által adott C lefogásra teljesül*

$$\frac{|C|}{OPT(G)} \leq 2 - \frac{8m}{13n^2 + 8m} \tag{3}$$

Bizonyítás. Legyenek $K_1, \ldots K_p$ a páratlan körök, amelyeket az algoritmus választott, legyen $k_i = |V(K_i)|$, és G_i az a gráf, amelyet G-ből kaptunk $V(K_1) \cup \ldots V(K_i)$ törlésével. Nyilván az algoritmus által adott C lefogásra $|C| = k_1 + \cdots + k_p + OPT(G_p)$. Az is látható, hogy

$$\begin{aligned} OPT(G) &\geq OPT(K_1) + \cdots + OPT(K_p) + OPT(G_p) \text{ (a 6.5 lemma alapján)} \\ &= \frac{k_1 + 1}{2} + \ldots \frac{k_p + 1}{2} + OPT(G_p) \quad \text{(mivel } K_i\text{-k páratlan körök).} \end{aligned} \tag{4}$$

Tekintsük a végső G_p gráfot, amely kielégíti vagy (a)-t, vagy (b)-t, ahol $n_p = |V(G_p)|$ és $m_p = |E(G_p)|$. A 6.2 lemma szerint legfeljebb $3(n - k_i)$ él van K_i és G_i között. Ezért $m_p \geq m - 3pn$.

A következő vizsgálatokban használjuk a tényt, hogy $a > b \geq 0$ és $c > 0$-ra teljesül $\frac{b}{a} < \frac{b+c}{a+c}$.

$$\begin{aligned} \frac{|C|}{OPT(G)} &\leq \frac{\sum_{i=1}^{p} k_i + OPT(G_p)}{\frac{1}{2}\sum_{i=1}^{p}(k_i + 1) + OPT(G_p)} = 2 - \frac{p + OPT(G_p)}{\frac{1}{2}\sum_{i=1}^{p}(k_i + 1) + OPT(G_p)} \\ &= 2 - \frac{2p + 2OPT(G_p)}{(n - n_p + p) + 2OPT(G_p)} \qquad (5) \\ &< 2 - \frac{2p}{n - n_p + p}. \qquad (6) \end{aligned}$$

Ezen a ponton már látszik, hogy az algoritmus 2-approximáció.

Ha $p \geq 4m/13n$, akkor kapjuk:

$$\frac{2p}{n - n_p + p} \geq \frac{8m/13}{n + 4m/13n} \geq \frac{8m}{13n^2 + 8m}.$$

Így a (6) alapján megkaptuk (3)-t. Ezért most felthetjük, hogy $p < 4m/13n$. $m_p \geq m - 3pn$ szerint $m_p \geq m - 12m/13 = m/13$. Vizsgáljuk meg külön azt a két esetet, amikor a while ciklus az (a) illetve a (b) miatt ér véget.

(a) eset: G_p egyetlen páratlan körből és néhány izolált pontból áll. Ekkor nyilván $n \geq k_1 + n_p \geq 6$, és $OPT(G_p) \geq m_p/2$. Mivel $m_p \geq m/13$, ezért $OPT(G_p) \geq m_p/2 \geq m/26$. Így

$$\frac{2p + 2OPT(G_p)}{(n - n_p + p) + 2OPT(G_p)} \geq \frac{2OPT(G_p)}{(n - n_p) + 2OPT(G_p)} \geq \frac{m/13}{n - n_p + m/13},$$

ami legfeljebb

$$\frac{m/13}{n + m/13} = \frac{m}{13n + m} \geq \frac{8m}{13n^2 + 8m}, \quad n \geq 8\text{-ra}.$$

Ha $6 \leq n \leq 7$, akkor ($n_p \geq 3$ miatt) $n - n_p \leq 4$. Így $n - n_p \leq 5n/8$ és

$$\frac{m/13}{n - n_p + m/13} \geq \frac{m/13}{5n/8 + m/13} = \frac{8m}{13 \cdot 5n + 8m} \geq \frac{8m}{13n^2 + 8m}.$$

(5)-ből megkapjuk (3)-at.

(b) eset: G_p páros gráf. Ekkor a 6.4 lemma alapján tudjuk, hogy $OPT(G_p) \geq m_p/n_p$. Mivel $m_p \geq m/13$, ezért $m_p/n_p \geq m/13n_p$. Így

$$\begin{aligned}
\frac{2p + 2OPT(G_p)}{(n - n_p + p) + 2OPT(G_p)} &\geq \frac{2OPT(G_p)}{(n - n_p) + 2OPT(G_p)} \geq \frac{2m_p/n_p}{(n - n_p) + 2m_p/n_p} \\
&\geq \frac{2(m/13n_p)}{(n - n_p) + 2(m/13n_p)} = \frac{2(m/13)}{(n - n_p)n_p + 2(m/13)} \\
&\geq \frac{2m/13}{n^2/4 + 2m/13} \qquad ((n - n_p)n_p \leq n^2/4 \text{ alapján}) \\
&= \frac{8m}{13n^2 + 8m}.
\end{aligned}$$

Ebből (5) alapján következik (3). □

6.3. Szabályos körök keresése

Az algoritmus a while ciklus magjában választ egy K szabályos páratlan kört. Az 6.2 lemma szerint egy legrövidebb páratlan kör egyben szabályos is. A következő lemma azt mutatja, hogy lineáris időben tudunk találni egy szabályos páratlan kört anélkül, hogy egy legrövidebb páratlan kört keresnénk.

6.7. Lemma. *Egy szabályos és húrmentes kör $O(n+m)$ időben megtalálható egy nem páros gráfban.*

Bizonyítás. Válasszuk ki G-nek egy összefüggő H komponensét, amely nem páros. Ez szélességi kereséssel $O(n+m)$ időben megtehető. Ha a szélességi keresést egy $s \in V(H)$ csúccsal kezdtük, particionáljuk $V(H)$-t $\{V_0, V_1, \dots V_r\}$-re, ahol V_i jelöli az s-től i távolságra lévő pontok halmazát. Nyilván $V_0 = \{s\}$. Legyen $k \in \{1, \dots r\}$ az a legkisebb index, amelyre $H[V_k]$ tartalmaz egy $e_k = (u_k v_k)$ élt. Ilyen él létezik, hiszen H nem páros. Ellenőrizzük le, hogy van-e olyan háromszög H-ban, amely e_k-t tartalmazza, $O(|V(H)| + |E(H)|)$ időben. Ha van ilyen háromszög, akkor mivel ez egy szabályos kör, készen vagyunk.

Tehát tegyük fel, hogy nincs e_k-t tartalmazó háromszög. Legyenek P_u és P_v az s-ből u_k-ba illetve v_k-ba vezető utak, $V(P_u) = \{u_0, u_1, \dots, u_k\}$, $V(P_v) = \{v_0, v_1, \dots, v_k\}$, $u_i, v_i \in V_i$, $0 \leq i \leq k$ (esetleg $u_i = v_i$ valamely i-re). Legyen $h \in \{0, 1, \dots, k-1\}$ a legnagyobb index, hogy V_h tartalmaz egy olyan s' csúcsot, amely szomszédos u_{h+1}-gyel és v_{h+1}-gyel, ahol $u_{h+1} \neq v_{h+1}$. Ilyen s' létezik, mivel $u_0 = v_0 = s$. Az $\{s', u_{h+1}, \dots, u_k\}$ és $\{s', v_{h+1}, \dots, v_k\}$ utak az e_k éllel egy K páratlan kört alkotnak. Megmutatjuk, hogy K szabályos és húrmentes. Ehhez tekintsünk egy $w \in V_i - V(K)$ csúcsot. Ha $i \leq h$, akkor w csak s'-vel, u_{h+1}-gyel valamint v_{h+1}-gyel lehet szomszédos $V(K)$-ból. Ha $h+1 \leq i \leq k-1$, akkor w nem lehet szomszédos sem u_i-vel, sem v_i-vel $V_i \cap V(K)$-ból a k választása miatt, $(V_0 \cup \dots \cup V_{i-1}) \cap V(K)$-ból csak u_{i-1}-gyel és v_{i-1}-gyel, $(V_{i+1} \cup \dots \cup V_k) \cap V(K)$-ból pedig u_{i+1} és v_{i+1} legfeljebb egyikével lehet szomszédos h választása miatt. Az utolsó eset, ha $i \geq k$, ekkor w a $(V_0 \cup \dots \cup V_{k-1}) \cap V(K)$-ból csak u_{k-1}-gyel és v_{k-1}-gyel,

$V_k \cap V(K)$-ból pedig u_k és v_k legfeljebb egyikével lehet szomszédos, mivel nincs e_k-t tartalmazó háromszög. Mindegyik esetben w-nek legfeljebb három szomszédja lehet $V(K)$-ból, azaz K szabályos. k és s' választásából pedig könnyen látható, hogy K húrmentes.

Az is látszik, hogy egy ilyen K kör megkonstruálása $O(n+m)$ időben lehetséges, mivel egy $w \in V(H)$ csúcsra illeszkedő élek halmazát $O(1)$ időben végignézhetjük. □

Mivel a while ciklusban legfeljebb n iteráció lehet, mindent összevetve azt kapjuk, hogy az algoritmus futási ideje $O(nm)$.

7. LP alsó becslés és lefogó ponthalmaz keresése

A minimális lefogó ponthalmaz feladat megfogalmazható IP feladatként. Legyen A a gráf incidenciamátrixa, azaz olyan $(m \times n)$-es mátrix, ahol az oszlopoknak pontok, a soroknak pedig élek felelnek meg. Vagyis

$$a_{ij} = \begin{cases} 1, & \text{ha } e_i \text{ illeszkedik } v_j\text{-re} \\ 0, & \text{különben.} \end{cases}$$

Legyen $\mathbf{1_k}$ a csupa 1-esből álló k dimenziós oszlopvektor. Ekkor az IP feladat a következőképpen néz ki:

$$Ax \geq \mathbf{1_m}$$
$$x \in \{0,1\}^n$$
$$\min \mathbf{1_n}^\top x$$

Vegyük ennek a feladatnak az LP relaxáltját.

$$Ax \geq \mathbf{1_m}$$
$$0 \leq x \leq \mathbf{1_n}$$
$$\min \mathbf{1_n}^\top x$$

Az LP relaxált x^* megoldása alsó korlátot ad az IP feladatra, $\mathbf{1_n}^\top x^* \leq \mathbf{1_n}^\top x^{IP}$. Ez hasznos lesz akkor, amikor a különböző approximációs algoritmusok eredményeit vizsgáljuk. Ugyanis ha egy tetszőleges algoritmus megoldása eléri az LP alsó korlátot vagy annak felső egészrészét, akkor biztosak lehetünk abban, hogy optimális megoldást kaptunk.

Emellett érdemes azt is megvizsgálni, hogy az LP feladat megoldásából, amely nem feltétlenül egész, hogyan kaphatunk lefogó ponthalmazt. Tehát adott egy $x^* = (x_1, \ldots, x_n)^\top$ vektor, ekkor legyen $C = \{v_i : x_i \geq \frac{1}{2}\}$.

7.1. Tétel. *C lefogó ponthalmaz, és mérete legfeljebb $2 \cdot OPT$.*

Bizonyítás. Ha C nem lenne lefogó ponthalmaz, akkor lenne olyan $e = (v_i v_j)$ él, amelyre $x_i < \frac{1}{2}$, $x_j < \frac{1}{2}$, ellentmondásban az $x_i + x_j \geq 1$ feltétellel. Továbbá a C incidenciavektora $y = [x^*]$, ahol $y_i \leq 2x_i^*$, így

$$OPT \leq \sum_{i=1}^{n} y_i \leq 2 \sum_{i=1}^{n} x_i^* \leq 2 \cdot OPT.$$

□

A fenti LP feladatnak mindig van félegész bázismegoldása, azaz olyan, hogy $x^* \in \{0, \frac{1}{2}, 1\}^n$. A szimplex algoritmus egy ilyet talál meg. (Ennél gyorsabb algoritmusok is ismertek.) Amennyiben sok koordinátában van $\frac{1}{2}$, akkor a kerekítéses módszer nagyon rossz megoldást adhat az optimálishoz képest.

A minimális lefogó ponthalmaz feladatot megfogalmazhatjuk hipergráfokra is. Ekkor az IP feladat és az LP relaxált ugyanúgy néznek ki, mint az előző esetben, természetesen az A mátrix sorai most 2-nél több 1-est tartalmahatnak, de legfeljebb f darabot. Most a lefogó ponthamaz legyen a következő: $C = \{v_i : x_i \geq \frac{1}{f}\}$. Erről a következőt mondhatjuk:

7.2. Tétel. *C lefogó ponthalmaz, és mérete legfeljebb $f \cdot OPT$.*

A bizonyítás ugyanúgy működik, mint az előző esetben.

8. Az algoritmusok gyakorlati összehasonlítása

Az előzőekben láthattuk, hogy az egyes algoritmusok milyen approximációs faktort tudnak biztosítani. Néhány helyen láttunk arra is példát, hogy az adott faktor éles. Felmerül a kérdés, hogy általában milyen eredményt tudnak adni ezek az algoritmusok. Ezért néhány algoritmust beprogramoztam MATLAB-ban. Kétféle gráfgeneráló algoritmust használtam.

- G1: Adott n pont, és bármely 2 pont között p valószínűséggel megy él. Ekkor a fokok eloszlása egyenletes.
- G2: Kiindulok egy kis gráfból, és ehhez egyesével veszek új pontokat. Minden új pontból adott számú meglévő pontba húzok élt úgy, hogy a már meglévő pontok közül az aktuális fokuk négyzetével arányos valószínűséggel választok. Ekkor előfordulhatnak nagyon kicsi és nagyon nagy fokú pontok is.

Kevés pontú gráfokra az LP feladatot is megoldottam, így megoldást és alsó korlátot is kaptam, de mivel a MATLAB szimplex algoritmust használ, és az lassú, így nagyobb gráfokra ezt már nem tudtam megtenni.

Jelölések:

- M - Mohó algoritmus
- P - Párosítás végpontjai
- DFS - Dfs fa nem-levelei
- LP - LP relaxált kerekítése
- AB - Alsó becslés az LP relaxáltból

Lássunk néhány futási eredményt 10, 50, 100, 500 és 1000 pontú gráfokra:

10	G1					G2				
M	2	5	2	5	3	6	6	7	5	6
P	2	5	2	6	3	6	7	7	5	6
DFS	4	6	2	6	4	7	7	8	6	6
LP	2	10	2	10	3	10	10	10	10	10
AB	2	5	2	5	3	5	5	5	5	5

50	G1					G2				
M	34	32	36	33	33	40	36	38	31	27
P	35	34	37	35	32	40	37	38	32	29
DFS	39	40	41	38	38	42	44	44	39	35
LP	50	50	48	50	48	50	50	50	49	49
AB	25	25	24,5	25	24,5	25	25	25	24,5	24,5

100	G1					G2				
M	76	75	77	76	77	70	73	70	71	72
P	77	76	77	78	78	72	76	71	72	74
DFS	83	84	86	82	84	81	89	89	86	86

500	G1					G2				
M	421	422	335	421	412	422	425	405	393	359
P	423	426	340	425	416	431	430	416	407	374
DFS	441	440	351	438	437	478	475	470	462	443

1000	G1					G2				
M	860	856	598	717	718	851	850	849	852	846
P	865	861	632	744	749	859	856	858	863	848
DFS	886	883	807	911	876	886	882	885	891	887

Amint látszik, a legtöbb esetben a mohó algoritmus adta a legjobb eredményt. (Ez érdekes, ha figyelembe vesszük, hogy az elméleti eredménye ennek az algoritmusnak a legrosszabb.) Ennél valamelyest rosszabb a párosításos

algoritmus, és sok esetben lényegesen rosszabb a dfs. Az LP relaxáltból számolt kerekített eredmény legtöbb esetben az alsó korlát kétszeresét adta, vagyis a várakozásainknak megfelelően rossz eredményt.

Az algoritmusok egy kicsit javíhatók a következő meggondolás alapján: ha van elsőfokú pont a gráfban, akkor annak egyetlen szomszédját bevéve a lefogó ponthalmazba biztos nem rontunk a lefogó ponthalmaz méretén. Ezért érdemes az első lépésben (és a dfs kivételével minden további iteráció előtt) az elsőfokú pontok szomszédait beválasztani a lefogásba és törölni a gráfból, amíg csak van elsőfokú pont. Ha nincs elsőfokú pont, akkor az algoritmus következő lépését hajtsuk végre, ekkor persze ismét keletkezhetnek elsőfokú pontok.

Súlyozott eset

9. Mohó algoritmus

9.1. Mohó algoritmus gráfokon

A mohó algoritmus mindig kiválasztja a legjobb pontot, azaz azt a pontot, amelyikre a $\min\{w(v)/d(v)\}$ érték felvétetik. Ezt a pontot beválasztja a lefogó ponthalmazba, majd törli a gráfból. Ezt iterálja mindaddig, amíg az összes élt le nem fogtuk.

9.2. Mohó algoritmus a halmazfedési feladatra

A mohó algoritmust általánosíthatjuk a halmazfedési feladatra. Ezt az algoritmust Chvátal [6] vizsgálta meg. Egy iterációban legyen C azon elemek halmaza, amelyeket már lefedtünk. Egy S halmaz *költség-hatékonyságának* nevezzük a $w(S)/|S-C|$ hányadost. Minden iterációban a leginkább költséghatékony halmazt választjuk.

Az algoritmusunk tehát a következő:

9.1. Algoritmus.

$C := \emptyset$, $J := \emptyset$;
while $C \neq U$ **do**
    $S_j := \text{argmin}\{w(S)/|S - C| : S \in \mathcal{S}\}$;
    $J := J \cup \{j\}$;
    $C := C \cup S_j$;
    $\mathcal{S} := \mathcal{S} - S_j$;
end
Output: J.

9.3. Az approximációs faktor

9.1. Tétel. *A mohó algoritmus $H(m)$-approximáció a minimális halmazfedés problémára, ahol $H(m) = 1 + \frac{1}{2} + \cdots + \frac{1}{m}$.*

Bizonyítás. Az egyszerűség kedvéért legyen $w_j = w(S_j)\ \forall j = 1, \ldots, n$. Legyen $A = (a_{ij})$ $m \times n$-es mátrix, ahol

$$a_{ij} = \begin{cases} 1, \text{ ha } e_i \in S_j \\ 0, \text{ különben.} \end{cases}$$

A halmazfedések incidenciavektorai, $x = (x_{ij})$ kielégítik az alábbi egyenlőtlenségrendszert:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i,$$

$$x_j \geq 0\ \forall j.$$

Vezessünk be nemnegatív változókat: $y_1, \ldots, y_m$, amelyek kielégítik a következőket:

$$\sum_{i=1}^{m} a_{ij} y_i \leq H(\sum_{i=1}^{m} a_{ij}) w_j \quad \forall j, \tag{7}$$

$$\sum_{i=1}^{m} y_i = \sum_{j \in J} w_j. \tag{8}$$

Vizsgáljuk meg ezeket a változókat: y_i azt jelenti, hogy mi a költsége e_i fedésének. Legyen t az iterációk száma (azaz $t = |J|$ az algoritmus végén), és $S_j^r := S_j - C$ az r-edik iteráció elején. Feltehetjük, hogy r iteráció után $J = \{1, \ldots, r\}$. Ekkor $|S_r^r|/w_r \geq |S_j^r|/w_j\ \forall r, j$. Minden e_i pontosan egy S_r^r-hez tartozik, ahol $r = 1, \ldots, t$ lehet. Erre az r-re $y_i = w_r/|S_r^r|$. Ekkor

$$\sum_{i=1}^{m} y_i = \sum_{r=1}^{t} \sum \{y_i : e_i \in S_r^r\} = \sum_{r=1}^{t} |S_r^r|(w_r/|S_r^r|) = \sum_{r=1}^{t} w_r = \sum_{j \in J} w_j$$

Ezzel megkaptuk a (8) egyenlőséget.

Legyen j rögztett. Figyeljük meg, hogy $S_j \cap S_r^r = S_j^r - S_j^{r+1}$. Legyen s a legnagyobb index, amelyre $|S_j^s| > 0$. Így

$$\sum_{i=1}^{m} a_{ij}y_i = \sum_{r=1}^{t}\sum\{y_i : e_i \in S_j \cap S_r^r\} = \sum_{r=1}^{t}(|S_j^r| - |S_j^{r+1}|)\cdot(w_r/|S_r^r|)$$
$$= \sum_{r=1}^{s}(|S_j^r| - |S_j^{r+1}|)\cdot(w_r/|S_r^r|) \leq w_j \sum_{r=1}^{s}(|S_j^r| - |S_j^{r+1}|)/|S_j^r|$$

Végül már csak a következőre van szükségünk:

$$\sum_{r=1}^{s}(|S_j^r| - |S_j^{r+1}|)/|S_j^r| \leq \sum_{r=1}^{s}(H(|S_j^r|) - H(|S_j^{r+1}|)) = H(|S_j^1|)$$
$$= H(|S_j|) = H(\sum_{i=1}^{m} a_{ij}).$$

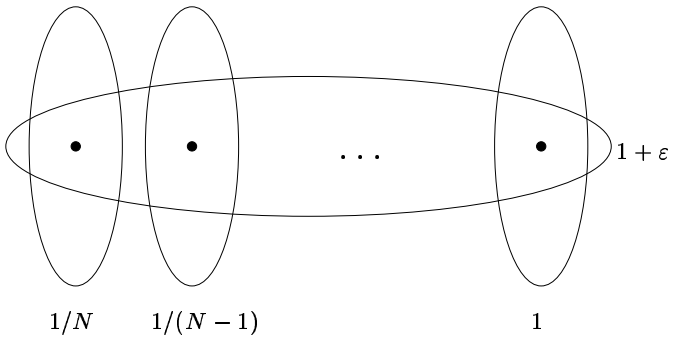
Ezeket összevetve megkapjuk a (7) egyenőtlenséget.

Az így kapott y változók segítségével a követezőt kapjuk:

$$\sum_{j=1}^{n} H(\sum_{i=1}^{m} a_{ij})w_j x_j \geq \sum_{j=1}^{n}(\sum_{i=1}^{m} a_{ij}y_i)x_j = \sum_{i=1}^{m}(\sum_{j=1}^{n} a_{ij}x_j)y_i \geq \sum_{i=1}^{m} y_i = \sum_{j\in J} w_j.$$

Az elejét és a végét összevetve, valamint a $H(\sum_{i=1}^{m} a_{ij}) \leq H(m)$ egyenlőtlenséget tekintve, ha x egy optimális fedés incidenciavektora, éppen a kívánt eredményt kapjuk. Sőt egy kicsivel erősebb eredményt is kapunk, mivel a $H(\sum_{i=1}^{m} a_{ij}) \leq H(N)$ egyenlőtlenség is igaz, ahol N a legnagyobb halmaz elemszáma, vagyis $\sum\{w_j : j \in J\} \leq H(N)\cdot OPT$. □

A fenti eredmény éles, amint azt a következő példa is mutatja:



Az algoritmus sorra beválasztja az egyelemű halmazokat, és így egy $H(N)$ súlyú fedést kapunk. Viszont egy optimális fedés súlya $1 + \varepsilon$.

10. Mohó algoritmus javítása

10.1. A mohó algoritmus javítása gráfokon

Amint az előző részben láttuk, az egyszerű mohó algoritmus approximációs faktora $O(\ln m)$. Clarkson [7] a következő egyszerű kis módosítást hajtotta végre: amikor megkapjuk a min=$\min\{w(v)/d(v)\}$ értéket egy v ponton, akkor v szomszédainak a súlyát csökkentjük min-nel. Amint azt látni fogjuk, ez az apró kis módosítás 2-approximációs algoritmust eredményez.

Lássuk az algoritmust formálisan:

10.1. Algoritmus.

$C := \emptyset$;
for $v \in V$ **do**
    $w_0(v) := w(v),\ d_0(v) := d(v)$;
end
for $e \in E$ **do**
    $c(e) := 0$;
end
while $\max\{d_0(v) : v \in V\} \geq 0$ **do**
    $v_i := \text{argmin}\{w_0(v)/d_0(v) : v \in V\}$;
    $min := w_0(v_i)/d_0(v_i)$;
    $C := C \cup \{v_i\}$;
    for $u :\ (uv_i) \in E$ **do**
        $w_0(u) := w_0(u) - min,\ c(uv_i) := min$;
    end
    $V := V - v_i$;
end
Output: C.

10.2. A javított approximációs faktor

10.1. Tétel. *A mohó algoritmus javított változata 2-approximáció.*

Bizonyítás. Vegyük észre, hogy $w_0(v) \geq 0$ és $c(e) \geq 0$ végig fennáll minden $v \in V$ csúcs és minden $e \in E$ él esetén. Továbbá a while ciklus elején fennáll $w(v) = w_0(v) + \sum\{c(uv) : (uv) \in E\}$. Ehhez csak annyit kell látnunk, hogy minden u csúcsra, amelyet érint a belső ciklus, a rá illeszkedő (uv) élt is érinti a belső ciklus, méghozzá úgy, hogy u súlyát csökkentjük, míg (uv) súlyát növeljük a min értékkel. A v_i csúcs súlyát $min \cdot d_0(v_i)$-vel csökkentjük, és pont $d_0(v_i)$ darab belőle kimenő él súlyát növeljük min-nel. Ebből következik:

$$w(v) \geq \sum_{(uv)\in E} c(uv) \quad \forall v \in V, \text{ és}$$
$$w(v) = \sum_{(uv)\in E} c(uv) \quad \forall v \in C.$$

Ezek segítségével könnyen látható:

$$\sum_{e\in E} c(e) \leq \sum_{v\in C^*} \sum_{(uv)\in E} c(uv) \leq \sum_{v\in C^*} w(v) = w(C^*), \quad \text{valamint}$$
$$w(C) = \sum_{v\in C} w(v) = \sum_{v\in C} \sum_{(uv)\in E} c(uv) \leq 2\sum_{e\in E} c(e).$$

Figyelembe véve, hogy $w(C^*) \leq w(C)$, az előzőekből $w(C)/w(C^*) \leq 2$-t kapjuk. □

11. Szintezés

11.1. A szintezéses technika

A csúcsokon értelmezett súlyfüggvényt *foksúlyozottnak* nevezzük, ha van olyan K konstans, hogy minden v csúcs súlya $K \cdot d(v)$. A most következő algoritmus lényege, hogy az adott w súlyfüggvényt foksúlyozott függvényekre bontjuk. Minden iterációban kiszámoljuk a legnagyobb foksúlyozott függvényt w-ben: $K = \min\{w(v)/d(v)\}$, $t(v) = K \cdot d(v)$. Ezt a t-t kivonva w-ből, a 0 súlyú pontokat beválasztjuk a lefogó ponthalmazba, majd a többi ponton a maradék súlyfüggvényre iteráljuk az eljárást.

Tehát az algoritmus a következő:

11.1. Algoritmus.

$C := \emptyset$;
for $v \in V$ **do**
    $w_0(v) := w(v),\ d_0(v) := d(v)$;
end
while nem üres a gráf **do**
        $K := \min\{w_0(v)/d_0(v) : v \in V\}$;
        for $v \in V$ **do**
            $t(v) := K \cdot d_0(v),\ w_0(v) := w_0(v) - t(v)$;
        end
        $W := \{v : w_0(v) = 0\}$;
        $C := C \cup W$;
        $V := V - W$;
end
Output: C.

11.2. Az approximációs faktor

11.1. Lemma. *Legyen $t : V \to \mathbb{R}_+$ foksúlyozott függvény. Ekkor teljesül a következő: $t(V) \leq 2 \cdot OPT$.*

Bizonyítás. Legyen K az a konstans, amelyre $t(v) = Kd(v)$, továbbá legyen C^* egy optimális lefogó ponthalmaz G-ben. Mivel C^* lefog minden élt, ezért $\sum\{d(v) : v \in C^*\} \geq |E|$, és így $t(C^*) \geq K|E|$. Továbbá tudjuk azt, hogy $\sum\{d(v) : v \in V\} = 2|E|$, innen $t(V) = 2K|E| \leq 2t(C^*) = 2OPT$. □

11.2. Tétel. *A szintezéses algoritmus 2-approximáció.*

Bizonyítás. Jelölje D_i az i-edik iterációban izolálttá vált pontok halmazát, $G_i = (V_i, E_i)$ az i-edik iteráció és D_i elhagyása után a maradék gráfot, k az iterációk számát.

Először belátjuk, hogy C lefogó ponthalmaz. Ha nem így lenne, akkor lenne olyan (uv) él, amelyre $u \in D_i$, $v \in D_j$ valamely i, j-re. Feltehető, hogy $i \leq j$. Ekkor $(uv) \in E_i$, ellentmondásban azzal, hogy u izolált.

Legyen C^* egy optimális lefogó ponthalmaz. Tekintsünk egy $v \in C$ pontot. Ha $v \in W_j$, akkor a súlyát a következőképpen tudjuk felbontani: $w(v) = \sum_{i \leq j} t_i(v)$. Most tekintsünk egy $v \in V - C$ pontot. Ha $v \in D_j$, akkor a súlyára alsó korlát: $w(v) \geq \sum_{i<j} t_i(v)$. Figyeljük meg, hogy az i-edik iterció után $C^* \cap G_i$ lefogó ponthalmaz G_i-ben. Így a (11.1) Lemma szerint $t_i(C \cap G_i) \leq 2t_i(C^* \cap G_i)$. A súlyok felbontása szerint:

$$w(C) = \sum_{i=1}^{k} t_i(C \cap G_i) \leq 2 \sum_{i=1}^{k} t_i(C^* \cap G_i) \leq 2w(C^*).$$

□

Az approximációs faktor nem javítható, mint azt a következő példa is mutatja: Legyen $G = K_{n,n}$ a teljes páros gráf $2n$ ponton, $w(v) = 1 \ \forall v$. A szintező algoritmus mind a $2n$ csúcsot beválasztja a lefogó ponthalmazba, holott egy optimális lefogásban elég csak az egyik oldal.

12. Lefogó ponthalmaz és általánosított párosítás

12.1. Általánosított párosítás

A súlyozatlan esetnél láttuk, hogy hogyan lehet egy párosításból megkapni egy lefogó ponthalmazt. Ennek egy általánosítása Gonzalez [8] algoritmusa.

Általánosított párosítás alatt az élek olyan $c : E \to \mathbb{R}$ súlyozását értjük, amelyre fennáll $w(v) \geq \sum\{c(e) : v \in e\}$ minden v csúcsra. Egy általánosított párosítás *maximális*, ha bármely él súlyát bármilyen $\delta > 0$ értékkel növelve már nem lesz általánosított párosítás. Egy v csúcsot *telítettnek* nevezünk, ha $w(v) = \sum\{c(e) : v \in e\}$ teljesül. Nyilván egy maximális általánosított párosításban nincs olyan él, amelynek egyik végpontja sem telített.

Egy maximális általánosított párosítás a következőképpen állítható elő: jelölje $tot(v)$ a v csúcsra illeszkedő élek súlyának összegét. Az élek kezdetben

0 súlyúak. Minden olyan élnek a súlyát, amelynek egyik végpontja sem telített még, növeljük, ameddig csak lehet. Azaz addig növelünk, míg valamelyik él egyik végpontja, v telítettté válik, vagyis $w(v) = tot(v)$ Ezután már csak a maradék éleken növelünk. Ezt addig folytatjuk, amíg végül minden élnek legalább az egyik végpontja telített.

Mivel a telített pontok lefogó ponthalmazt alkotnak, így a lefogó ponthalmaz kereső algoritmus a következő:

12.1. Algoritmus.

$T := \emptyset$;
for $v \in V$ **do**
    $tot(v) := 0$;
end
for $e \in E$ **do**
    $c(e) := 0$;
end
while $E \neq \emptyset$ **do**
    $min := \min\{(w(v) - tot(v))/d(v) : v \in V\}$;
    for $e \in E$ **do**
        $c(e) := c(e) + min$;
    end
    for $v \in V$ **do**
        $tot(v) := tot(v) + d(v) \cdot min$;
    end
    $T := \{v : w(v) = tot(v)\}$;
    $V := V - T$;
    $E := E - \{e = (uv) : u \in T \text{ or } v \in T\}$;
end
Output: T.

12.2. Az approximációs faktor

12.1. Tétel. *A telített pontok súlya legfeljebb* $2 \cdot OPT$.

Bizonyítás. Legyen C^* egy optimális lefogás. Ennek segítségével bontsuk fel T-t két halmazra:

$$T(in) := T \cap C^*,\ T(out) := T - C^*.$$

Hasonlóan, a nem-telített pontok halmazát is bontsuk fel:

$$NT(in) := (V - T) \cap C^*,\ NT(out) := (V - T) - C^*.$$

Nyilván $OPT \geq w(T(in))$. Megmutatjuk, hogy $OPT \geq w(T(out))$. Mivel minden $v \in T(out)$ telített pont, ezért $w(T(out)) = c(E(out))$, ahol $E(out)$ jelöli a $T(out)$-beli pontokra illeszkedő élek halmazát. Mivel $T(out)$ diszjunkt C^*-tól, és C^* lefogó ponthalmaz, ezért $T(out)$ független halmaz, és a $T(out)$-beli pontok szomszédai C^*-ban vannak. Abból, hogy $w(v) \geq tot(v)\ \forall v \in V$, látható, hogy

$$c(E(out)) \leq w(T(in)) + w(NT(in)) = w(C^*) = OPT.$$

Ebből $w(T(out)) \leq OPT$, és így

$$w(T) = w(T(in)) + w(T(out)) \leq 2 \cdot OPT.$$

□

13. Az előző algoritmusok összehasonlítása

A szintezéses (SZ) algoritmus a pontok súlyát csökkenti, míg az általánosított párosításos (ÁP) algoritmus az élek súlyát növeli. Ez a növelés azonban a pontok súlyainak és a már meglévő élsúlyok függvényében történik. Felvetődik a kérdés, hogy ez a két módszer összefügg-e egymással, és ha igen, akkor hogyan. Megmutatjuk, hogy valójában ugyanarról van szó.

Ezt az iterációk száma szerinti indukcióval látjuk be. A jobb átláthatóság kedvéért vezessünk be néhány jelölést. Legyen t az iterációk száma. w az eredeti súlyfüggvény a pontokon, legyen $w_i^l(v)$ a v pont súlya az SZ i-edik iterációja után, min_i^l a $min = K$ érték az i-edik iteráció során. Az ÁP-nél pedig legyen $tot_i^a(v)$ a $tot(v)$ értéke az i-edik iterációja után, min_i^a a min érték az i-edik iteráció során. Megmutatjuk a következőt:

13.1. Állítás.

$$w(v) - tot_i^a(v) = w_i^l(v) \qquad \forall v \in V,\ \forall\ 1 \leq i \leq t.$$

13.2. Megjegyzés. *Ez azt jelenti, hogy az i-edik iterációban ugyanazok a pontok kerülnek be ÁP szerint T-be, mint SZ szerint W-be. Tehát a maradék pontok halmaza is ugyanaz a V_i lesz, és a fokfüggvény ugyanaz a d_i.*

Bizonyítás. Az első iterációban $min_1^l = min_1^a = \min\{w(v)/d(v) : v \in V\}$, így

$$w_1^l(v) = w(v) - d(v)min_1^l = w(v) - d(v)min_1^a = w(v) - tot_1^a(v) \quad \forall v \in V.$$

Tegyük fel, hogy valamely $1 \leq i \leq t$-re már beláttuk az állítást. Legyen az $i+1$-edik lépésben u^l az SZ szerinti, u^a az ÁP szerinti minimum helye. Ekkor

$$min_{i+1}^l = \frac{w_i^l(u^l)}{d_i(u^l)} = \frac{w(u^l) - tot_i^a(u^l)}{d_i(u^l)} \geq \frac{w(u^a) - tot_i^a(u^a)}{d_i(u^a)} = min_{i+1}^a,$$

valamint

$$min_{i+1}^a = \frac{w(u^a) - tot_i^a(u^a)}{d_i(u^a)} = \frac{w_i^l(u^a)}{d_i(u^a)} \geq \frac{w_i^l(u^l)}{d_i(u^l)} = min_{i+1}^l,$$

vagyis $min_{i+1}^l = min_{i+1}^a$. Így jelölhetjük a közös min_{i+1}-gyel. Tudjuk, hogy $tot_{i+1}^a(v) = tot_i^a(v) + d_i(v)min_{i+1}$, valamint $w_{i+1}^l(v) = w_i^l(v) - d_i(v)min_{i+1}$, és így

$$\begin{aligned} w(v) - tot_{i+1}^a(v) &= w(v) - tot_i^a(v) - d_i(v)min_{i+1} \\ &= w_i^l(v) - d_i(v)min_{i+1} = w_{i+1}^l(v) \qquad \forall v \in V_i. \end{aligned}$$

□

13.3. Következmény. *Az ÁP és az SZ algoritmus tetszőleges $G = (V, E)$ gráfra ugyanazt az eredményt adja.*

14. Élek, elemek egyenkénti vizsgálata

14.1. Élválasztásos algoritmus

Ez egy nagyon egyszerű algoritmus. Választunk egy tetszőleges élt, és ennek a két végpontját vizsgáljuk. Azt, amelyiknek kisebb a súlya, beválasztjuk a lefogó ponthalmazba, töröljük a gráfból, a másik pont súlyát pedig csökkentjük a törölt pont súlyával. Ezt iteráljuk addig, amíg még van él a gráfban.

Mint látni fogjuk, ez 2-approximációs algoritmus.

14.2. Általánosítás a halmazfedési problémára

A következő algoritmust Bar-Yehuda és Even [1] találták ki. Jelölje az e elemet taralmazó halmazok indexeinek halmazát $F(e)$. Ezzel a jelöléssel $f = \max\{|F(e)| : e \in U\}$. Válasszunk ki egy tetszőleges e elemet, és vizsgáljuk meg $F(e)$-beli indexű halmazokat. A minimális súlyút (legyen ez S_i) válasszuk be a halmazfedésbe, a többinek a súlyát pedig csökkentsük S_i súlyával. Töröljük U-ból az S_i-beli elemeket. Ezt iteráljuk addig, amíg van fedetlen elem.

Tehát az algoritmus a következőképpen néz ki:

14.1. Algoritmus.

$C := \emptyset,\ J := \emptyset;$
for $S \in \mathcal{S}$ **do**
    $w_0(S) := w(S);$
end
for $e \in U$ **do**
    $c(e) := 0;$

end

while $U \neq \emptyset$ **do**

Legyen $e \in U$;

$S_i := \text{argmin}\{w_0(S_k) : k \in F(e)\}$, $min := w_0(S_i)$;

$c(e) := min$;

for $k \in F(e)$ **do**

$w_0(S_k) := w_0(S_k) - min$;

end

$C := C \cup S_i$, $J := J \cup \{i\}$;

$U := U - S_i$;

end

Output: J.

14.3. Az algoritmus approximációs faktora

14.1. Tétel. *Az algoritmus által adott halmazfedés W súlyára teljesül a következő:*

$$W \leq OPT \cdot \max_{e \in U} |F(e) \cap J|.$$

Bizonyítás. Nyilván $w_0(S_i) \geq 0 \ \forall S_i \in \mathcal{S}$, és $c(e) \geq 0 \ \forall e \in U$ teljesül. Továbbá teljsül $w_0(S_i) + \sum\{c(e) : e \in S_i\} = w_i \ \forall S_i \in \mathcal{S}$ is. Ez kezdetben nyilván igaz, a továbbiakban pedig, amikor $e \in S_i$ súlya min-re nő, $w_0(S_i)$ min-nel csökken. Ebből látszik:

$$w_i \geq \sum_{e \in S_i} c(e) \quad \forall i \in I, \text{ és}$$

$$w_i = \sum_{e \in S_i} c(e) \quad \forall i \in J.$$

Tekintsük a következőt:

$$W = \sum_{i \in J} w_i = \sum_{i \in J} \sum_{e \in S_i} c(e).$$

Ebben a szummában $c(e)$-t pontosan $|F(e) \cap J|$-szer szálmoltuk, amiből

$$W \leq \sum_{e \in U} c(e) \cdot \max_{e \in U} |F(e) \cap J| \tag{9}$$

következik.

Vezessünk be új változókat, amelyek kielégítik a következőket:

$$\sum_{e \in S_i} y_e \leq w_i \quad \forall i \in I,$$

$$y_e \geq 0 \quad \forall e \in U.$$

$y_e = c(e)$ kielégíti ezeket a követelményeket. Tekintsük a duális programot:

$$\sum_{i \in F(e)} x_i \geq 1 \quad \forall e \in U,$$

$$x_i \geq 0 \quad \forall i \in I.$$

Minimalizáljuk a $\sum\{x_i w_i : i \in I\}$ értéket. Először tekintsük a következőt:

$$\sum_{e \in U} y_e \leq \sum_{e \in U} y_e \sum_{i \in F(e)} x_i = \sum_{i \in I} x_i \sum_{e \in S_i} y_e \leq \sum_{i \in I} x_i w_i.$$

Ebből következik, hogy

$$\sum_{e \in U} y_e \leq \min \sum_{i \in I} x_i w_i. \tag{10}$$

Ha J^* egy optimális fedés indexhalmaza, akkor legyen

$$x_i = 1, \qquad i \in J^*,$$

$$x_i = 0, \qquad i \notin J^*.$$

Ezek az x_i-k kielégítik a fenti követelményeket, és ezért

$$OPT \geq \min \sum_{i \in I} x_i w_i. \tag{11}$$

(10) és (11) segítségével

$$\sum_{e \in U} c(e) \leq OPT,$$

így (9) felhasználásával a tételt beláttuk. □

A tételből könnyen látszik a gyengébb állítás is, vagyis hogy $W \leq f \cdot OPT$. Ez éles, amint azt a következő példa is mutatja. Legyenek a halmazaink a következőek:

$$\begin{aligned} S_1 &= e_1, \\ S_2 &= \{e_1, e_2\}, \\ S_3 &= \{e_1, e_3\}, \\ &\vdots \\ S_{n-1} &= \{e_1, e_{n-1}\}, \\ S_n &= \{e_1, e_2, \ldots, e_n\}. \end{aligned}$$

Minden halmaz súlya legyen w. Ha az algoritmus először e_1-et választja, és e_n-t utoljára, akkor minden halmaz belekerül a fedésbe. Így $W = n \cdot w$, holott S_n egy optimális fedés, azaz $OPT = w$, és $f = n$.

15. A Local-ratio theorem

15.1. Egy lokális algoritmus

Bar-Yehuda és Even [2] eredménye a Local-ratio theorem. Ennek lényege, hogy a gráfnak egyszerre csak kis részeit vizsgálva építjük föl a lefogó ponthalmazt, és az approximációs faktort is ezen részek segítségével kapjuk meg.

Legyen most A egy tetszőleges approximációs algoritmus a minimális súlyú lefogó ponthalmaz problémára. Ha C_A az algoritmus által adott lefogó ponthalmaz, C^* pedig egy optimális, akkor definiáljuk a következő értéket: $R_A(G, w) = w(C_A)/w(C^*)$. Legyen $\bar{G}$ egy $\bar{n}$ pontú, súlyozatlan gráf, amelynek az optimális lefogó ponthalmaza $\bar{c}^*$ elemű. Legyen $\bar{r} = \bar{n}/\bar{c}^*$. A és $\bar{G}$ felhasználásával tekintsük a következő algoritmust:

15.1. Algoritmus ($LOCAL$).

Legyen $\tilde{G}(\tilde{V}, \tilde{E})$ a G egy részgráfja, amely izomorf $\bar{G}$-sal;
Legyen $0 \leq \delta \leq \min\{w(v) : v \in \tilde{V}\}$;

for $v \in V$ **do**
  if $v \in \tilde{V}$ **then**
    $w_0(v) := w(v) - \delta$;
  else
    $w_0(v) := w(w)$;
  end
end
$C_0 := A(G, w_0)$;
$C := C_0$;
Output: C.

15.1. Lemma. *Legyen G egy gráf w, w_1 és w_2 súlyfüggvényekkel, amelyekre teljesül, hogy minden $v \in V$: $w(v) \geq w_1(v) + w_2(v)$. Legyenek C^*, C_1^* és C_2^* optimális lefogó ponthalmazok a megfelelő súlyokhoz. Ekkor teljesül, hogy $w(C^*) \geq w_1(C_1^*) + w_2(C_2^*)$.*

Bizonyítás.

$$\begin{aligned} w(C^*) &= \sum_{v \in C^*} w(v) \geq \sum_{v \in C^*} (w_1(v) + w_2(v)) \\ &= w_1(C^*) + w_2(C^*) \geq w_1(C_1^*) + w_2(C_2^*) \end{aligned}$$

□

15.2. Tétel (Local-ratio theorem). $R_{LOCAL}(G, w) \leq \max\{\bar{r}, R_A(G, w_0)\}$

Bizonyítás. Legyenek c^* és c_0^* a w-hez illetve w_0-hoz tartozó optimális lefogások súlyai, valamint legyen $r = \max\{\bar{r}, R_A(G, w_0)\}$. Ekkor

$$\begin{aligned} w(C) &\leq w_0(C) + \delta \cdot \bar{n} && \text{(mivel } |C \cap \tilde{V}| \leq n) \\ &= R_A(G, w_0) \cdot c_0^* + \bar{r} \cdot \delta \cdot \bar{c}^* && \text{(a definíciókból)} \\ &\leq r \cdot (c_0^* + \delta \bar{c}^*) && (r \text{ definíciójából}) \\ &\leq r \cdot c^* && \text{(a lemmából).} \end{aligned}$$

□

15.2. Egy általánosabb algoritmus

Tekintsük most gráfoknak egy véges Γ családját, valamint legyen ehhez $r_\Gamma = \max\{\bar{r}_G : G \in \Gamma\}$. Egy $G = (V, E)$ gráfra és $U \subseteq V$ ponthalmazra jelölje $G[U]$ az U által feszített részgráfot. Tekintsük a következő algoritmust:

15.2. Algoritmus ($LOCAL_\Gamma$).

for $v \in V$ **do**
    $w_0(v) := w(w)$;
end
for $\tilde{G}(\tilde{V}, \tilde{E})$: G részgráfja, amely izomorf valamely $\bar{G} \in \Gamma$-val **do**
    $\delta := \min\{w_0(v) : v \in \tilde{V}\}$;
    for $v \in \tilde{V}$ **do**
        $w_0(v) := w_0(v) - \delta$;
    end
end
$C_1 := \{v : w_0(v) = 0\}$;
$V_1 := V - C_1$;
$C_2 := A(G[V_1], w_0)$;
$C := C_1 \cup C_2$;
Output: C.

15.3. Tétel. $R_{LOCAL_\Gamma}(G, w) \leq \max\{r_\Gamma, R_A(G[V_1], w_0)\}$

Bizonyítás. Indukcióval a második for ciklus iterációinak számára, i-re.

$i = 0$-ra az állítás triviális, $i = 1$-re pedig éppen a 15.2 tétel.

Tegyük fel, hogy az állítás igaz i-re, és valamely (G, w)-re a második ciklusban $i + 1$ iterációja van. Legyen $\bar{G} \in \Gamma$ az a gráf, amelyik az első iterációban szerepel. Ekkor tekinthetjük a $LOCAL$ algoritmust, amely a $\bar{G}$-sal izomorf részgráf kiválasztása után az $A = LOCAL_{\Gamma - \bar{G}}$-t hívja meg. Az indukció szerint kész vagyunk. □

15.4. Megjegyzés. *Ha Γ egyetlen él, és $A = LOCAL_\Gamma$, akkor az élválasztós algoritmust kapjuk, amelynek approximációs faktora a 15.3 tétel alapján 2.*

16. Egy párhuzamosított primál-duál módszer

16.1. Élpakolás, elempakolás

Khuller, Vishkin és Young [11] kifejlesztettek egy párhuzamos algoritmust a hipergráfos lefogó ponthalmaz problémára. Ehhez tekintették a duális feladatot. *Élpakolásnak* egy olyan $c : E \to \mathbb{R}_+$ hozzárendelést nevezünk, amelyre $c(E(v)) \leq w(v)$ minden v pontra, ahol $E(v)$ a v-re illeszkedő élek halmaza. (Ez a definíció hagyományos gráfok esetén egy az egyben megegyezik az általánosított párosítás definíciójával.) A maximális súlyú élpakolás probléma $c(E)$ maximalizálását jelenti.

A lefogó ponthalmaz és az élpakolás feladat relaxáltjai lineáris programozási duálisok.

A halmazfedési feladatot a bevezetőben már definiáltam. Ennek duális feladata az *elempakolási probléma*, egy olyan $c : U \to \mathbb{R}_+$ súlyfüggvény megkeresése, amelyre minden S halmazra a benne lévő elemek összsúlya legfeljebb $w(S)$.

Ahogy a halmazfedési feladatot megfeleltettük a hipergráfos lefogó ponthalmaz feladatnak, ugyanígy a duális problémák is megfeleltethetők egymásnak.

16.2. ϵ-maximális pakolás

Mostantól adott egy $\epsilon > 0$ szám. Bevezetésként nézzünk néhány lemmát.

16.1. Lemma. *Legyen C tetszőleges lefogás, c egy tetszőleges élpakolás. Ekkor $c(E) \leq w(C)$.*

Bizonyítás.

$$c(E) = \sum_{e \in E} c(e) \leq \sum_{e \in E} |e \cap C| c(e) = \sum_{v \in C} c(E(v)) \leq \sum_{v \in C} w(v) = w(C).$$

□

16.2. Lemma. *Legyen C egy lefogás, c egy élpakolás úgy, hogy teljesüljön $c(E(v)) \geq (1 - \epsilon) w(v)$ $\forall v \in C$. Ekkor $(1 - \epsilon) w(C) \leq f \cdot c(E)$.*

Bizonyítás.

$$(1-\epsilon)w(C) = (1-\epsilon)\sum_{v\in C} w(v) \leq \sum_{v\in C} c(E(v)) = \sum_{e\in E} |e\cap C|c(e) \leq f\cdot c(E).$$

□

16.3. Lemma. *A 16.2 lemmában, ha a súlyok egészek és $\epsilon < 1/w(V)$, akkor $w(C) \leq f\cdot OPT$.*

Bizonyítás. Legyen C^* egy minimális súlyú lefogás. A 16.2 lemma alapján $(1-\epsilon)w(C) \leq fw(C^*)$, így $w(C) \leq \lfloor fw(C^*) + \epsilon w(C)\rfloor = fw(C^*)$. □

Ha adott egy c pakolás, legyen $C_c = \{v \in V : c(E(v)) \geq (1-\epsilon)w(v)\}$. Ha C_c egy lefogás, akkor c ϵ-maximális. c pontosan akkor 0-maximális, ha maximális.

16.3. Az algoritmus

Most bemutatunk egy páruzamos algoritmust a minimális lefogó ponthalmaz problémára. Az előzőekben redukáltuk a problémát egy ϵ-maximális pakolására. Az algoritmus fenntart egy c pakolást és egy részleges lefogást: $C_c = \{v \in V : c(E(v)) \geq (1-\epsilon)w(v)\}$. Az algoritmus növeli a $c(e)$-ket addig, amíg c ϵ-maximális és C_c lefogás nem lesz. Amikor egy v csúcs bekerül C_c-be, v-t és az őt tartalmazó hiperéleket töröljük a hipergráfból. Jelölje E_c a maradék hiperélek halmazát, $E_c(v)$ a v-t tartalmazó maradék hiperélek halmazát, $d_c(v)$ a maradék fokot és $w_c(v)$ a maradék súlyt.

A while ciklus minden iterációjában $c(e)$-t akarjuk növelni a maradék e hiperélekre. Ahhoz, hogy c pakolás maradjon, minden $v \in e$ csúcs korlátozza $c(e)$ növelését $w_c(v)/d_c(v)$-vel. Az algoritmus alábbi leírásában c növelése helyett w_c csökkentése szerepel. Ahogy az SZ és ÁP algoritmusok esetén láttuk, hogy ez a kétfajta változtatás megfelel egymásnak, ez most is könnyen belátható.

16.1. Algoritmus.

$C := \emptyset$, $V_c = V$, $E_c := E$;
for $v \in V$ **par do**
    $w_c(v) := w(v),\ E_c(v) := E(v),\ d_c(v) := |E(v)|$;
end
while $E_c \neq \emptyset$ **do**
    for $e \in E_c$ **par do**
        $\delta(e) := \min\{w_c(v)/d_c(v) : v \in e\}$;
    end
    for $v \in V_c$ **par do**
        $w_c(v) := w_c(v) - \sum_{e \in E_c(v)} \delta(e)$;
        if $w_c(v) \leq \epsilon w(v)$ **then**
            $C := C \cup \{v\},\ V_c := V_c - \{v\}$;
        end
    end
end
Output: C.

Itt az egyes for ciklusokban párhuzamosan hajtjuk végre a műveleteket.

Az előző részben szerplő lemmák segítségével könnyen látszik a következő eredmény:

16.4. Tétel. *Az algoritmus $f/(1-\epsilon)$-appriximéciót ad. Vagyis hagyományos gráfokra $2/(1-\epsilon)$-approximáció.*

16.4. Bonyolultságvizsgálat

16.5. Tétel. *Az algoritmus $O(f \ln^2 m \ln \frac{1}{\epsilon})$ időt, $M/\ln^2 m$ processzort, azaz $O(fM \ln \frac{1}{\epsilon})$ műveletet igényel.*

Bizonyítás. Adott c pakoláshoz vezessük be a következő értéket:

$$\phi_c = \sum_{v \in V} d_c(v) \ln \frac{w_c(v)}{\epsilon w(v)}.$$

Vizsgáljuk meg, hogy ϕ_c hogyan változik a while ciklus egy iterációja alatt. Jelölje ϕ_i az i-edik iteráció előtti értéket, ϕ_{i+1} az i-edik iteráció utáni értéket. Hasonlóan bevezetjük a d_i, E_i, V_i jelöléseket is.

16.6. Lemma. $\phi_i - \phi_{i+1} \geq |E_{i+1}|$.

Bizonyítás. Az i-edik iteráció alatt azt mondjuk, hogy egy v pont *korlátoz* egy rá illeszkedő $e \in E_i$ hiperélt, ha a $\min_{v \in e} w_i(v)/d_i(v)$ érték v-n vétetik fel. Egy v pont $L(v)$ darab hiperélt korlátoz. Így $w_{i+1}(v) \leq w_i(v)(1-L(v)/d_i(v))$. Ekkor

$$\begin{aligned}\phi_i - \phi_{i+1} &= \sum_{v \in V_i} \left(d_i(v) \ln \frac{w_i(v)}{\epsilon w(v)} - d_{i+1}(v) \ln \frac{w_{i+1}(v)}{\epsilon w(v)} \right) = \sum_{v \in V_{i+1}} d_i(v) \ln \frac{w_i(v)}{\epsilon w(v)} \\ &\geq \sum_{v \in V_{i+1}} d_i(v) \ln \frac{w_i(v)}{w_{i+1}(v)} \geq \sum_{v \in V_{i+1}} -d_i(v) \ln(1 - L(v)/d_i(v)) \\ &\geq \sum_{v \in V_{i+1}} L(v) \geq |E_{i+1}|.\end{aligned}$$

Az utolsó előtti lépés a $-\ln(1-x) \geq x$ összefüggésből jön. Az utolsó lépés pedig abból következik, hogy minden megmaradó hiperélt korlátoz valamely $v \in V_{i+1}$ pont. □

16.7. Lemma. *Legfeljebb* $(1 + f \ln \frac{1}{\epsilon})(1 + \ln m)$ *iteráció van.*

Bizonyítás. Legyen $a = f \ln \frac{1}{\epsilon}$. Könnyen kiszámolható, hogy $\phi_{i+1} \leq |E_{i+1}|a$. Az előző lemma felhasználásával $\phi_{i+1} \leq \phi_i(1 - 1/(a+1))$. $\phi_1 \leq ma$, és indukcióval

$$\phi_i \leq ma(1 - 1/(a+1))^{i-1} \leq ma \exp(-(i-1)/(a+1)),$$

ahol felhasználtuk, hogy $e^x \geq 1 + x$. Rögzítsük le most i-t a következőnek: $i = 1 + \lceil (a+1) \ln m \rceil$, ekkor $\exp(-(i-1)/(a+1)) \leq \exp(-\ln m) = 1/m$, ezzel $\phi_i \leq a$. Az utolsó iteráció kivételével minden iteráció után marad legalább egy hiperél, ezért ϕ_c legalább eggyel csökken. Így $i + a = (1 + f \ln \frac{1}{\epsilon})(1 + \ln m)$ iteráció után $\phi_c \leq 0$. □

Ha a while ciklus i-edik iterációja elején $|E_i| = q$, akkor az iteráció $O(\ln q)$ időt igényel. Így a lemma alapján a teljes idő $O(f \ln^2 m \ln \frac{1}{\epsilon})$. Egy iterációban $O(fq)$ művelet van, emiatt a műveletek teljes száma f-szer az egyes iterációkban lévő élszámok összege. A 16.6 lemma szerint $\phi_i - \phi_{i+1} \geq |E_{i+1}|$, továbbá $\phi_0 = M \ln \frac{1}{\epsilon}$, így ($t$-vel jelölve az iterációk számát, vagyis $\phi_t = 0$):

$$|E_0| + |E_1| + \cdots + |E_t| \leq m + (\phi_0 - \phi_1) + \cdots + (\phi_{t-1} - \phi_t) = m + M \ln \frac{1}{\epsilon}.$$

Tehát összesen $O(fM \ln \frac{1}{\epsilon})$ művelet van. A műveleteket hatékonyan tudjuk ütemezni úgy, hogy az idő legfeljebb konstanssal nő. Így a műveletszám és az idő hányadosából megkapjuk a szükséges processzorok számát, vagyis $M/\ln^2 m$-et. □

17. Lefogó ponthalmaz keresése LP segítségével

Amint a súlyozatlan esetben láttuk, az LP relaxált segítségével könnyen tudunk megoldást találni a minimális lefogó ponthalmaz feladatra. Most adott egy $w : V \to \mathbb{R}_+$ súlyfüggvény. Legyen $w = (w_1, \ldots, w_n)^\top$, ahol $w_i = w(v_i)$. Nézzük meg, hogy ebben az esetben hogy néz ki az LP relaxált.

$$Ax \geq \mathbf{1_m}$$
$$0 \leq x \leq \mathbf{1_n}$$
$$\min w^\top x$$

Minden eredmény, amelyet a súlyozatlan esetben láttunk, egyszerű módón átmegy a súlyozott esetre is, ezért ezeket most nem részletezem.

Most nézzünk egy másik megközelítést, amellyel lineáris programozási módszerrel kaphatunk megoldást. Ezt most a halmazfedési feladatra fogalmazzuk meg. Tekintsük az LP duális feladatot.

$$y^\top A \leq w^\top$$
$$0 \leq y$$
$$\max \mathbf{1_m}^\top y$$

17.1. Definíció. *Egy $\bar{y}$ megengedett megoldásról azt mondjuk, hogy* maximális, *ha nincs olyan y megengedett megoldás, amelyre $\sum_{i=1}^m y_i > \sum_{i=1}^m \bar{y}_i$ és $y_i \geq \bar{y}_i$ teljesül.*

Legyen tehát most $\bar{y}$ egy maximális megengedett megoldása a duális feladatnak. Legyen $C = \{j : \sum_{i=1}^m a_{ij}\bar{y}_i = w_j\}$.

17.2. Tétel. *C megengedett megoldás a lefogó ponthalmaz feladatra, továbbá $w(C) \leq f \cdot OPT$.*

Bizonyítás. Először belátjuk, hogy C megengedett. Tegyük fel, hogy nem megengedett, ekkor van egy olyan g elem, amelyet nem fedtünk. Legyen $\delta = \min_{j:g\in S_j}\{w_j - \sum_{j=1}^m a_{ij}\bar{y}_i\} > 0$, $\mathbf{e_g}$ az az egységvektor, ahol a g elemnek megfelelő koordináta 1-es, a többi 0 . Ekkor $y = \bar{y} + \delta \cdot \mathbf{e_g}$ megengedett megoldás, ellentétben $\bar{y}$ maximalitásával.

A tétel második feléhez legyen x^* a primál LP feladat egy optimális megoldása. Ekkor tekintsük a következőt:

$$w(C) = \sum_{j\in C} w_j = \sum_{j\in C}(\sum_{i=1}^m a_{ij}\bar{y}_i) \leq \max_i(\sum_{j\in C} a_{ij}) \sum_{i=1}^m \bar{y}_i$$
$$\leq \max_i(\sum_{j\in C} a_{ij}) \sum_{j=1}^n w_j x_j^* = f \cdot OPT_{LP} \leq f \cdot OPT$$

□

18. Az algoritmusok gyakorlati összehasonlítása

A súlyozatlan esethez hasonlóan a súlyozott eset algoritmusait is megvizsgáltam konkrét futási eredmények alapján. Ugyanazt a két gráfgeneráló algoritmust használtam, és a pontokhoz véletlen súlyokat rendeltem.

A primál-duál algoritmust természetesen nem párhuzamosan programoztam be, a for ciklusok lépéseit egyenként hajtatom végre, de ez csak a futási idő növekedésével jár, a végeredményt nem befolyásolja.

Jelölések:

- M - Mohó algoritmus
- JM - Javított mohó algoritmus
- SZ - Szintezéses algoritmus
- EV - Élválasztásos algortimus
- PD - Primál-duál algoritmus
- LP - LP relaxált kerekítése
- AB - Alsó becslés az LP relaxáltból

Lássuk, hogy milyen eredmények születtek:

10	G1					G2				
M	28	34	14	33	32	18	26	30	32	29
JM	28	43	14	33	32	18	26	30	32	32
SZ	28	43	14	33	37	18	26	30	32	32
EV	28	37	14	33	32	18	26	30	26	29
PD	28	36	14	33	37	18	26	30	32	32
LP	49	29	14	51	45	18	26	49	47	28
AB	24,5	29	14	25,5	22,5	18	26	24,5	23,5	28

50	G1					G2				
M	184	181	198	175	173	148	154	141	179	170
JM	185	196	214	178	183	155	158	148	195	170
SZ	185	196	214	178	183	155	158	148	195	170
EV	187	191	196	193	195	165	180	161	186	189
PD	185	193	214	178	173	150	158	144	182	170
LP	249	244	264	241	257	213	238	232	251	251
AB	124,5	122	132	120,5	128,5	110,5	119,5	117	128	125,5

100	G1					G2				
M	376	311	356	416	381	376	334	327	371	371
JM	384	320	371	436	408	387	349	348	379	386
SZ	384	320	371	436	408	387	349	348	379	386
EV	409	342	386	458	432	414	353	351	401	413
PD	380	320	363	426	391	382	339	344	388	379

500	G1					G2				
M	2283	1979	1971	2379	2091	1144	579	842	1082	1282
JM	2379	2090	2109	2424	2185	1188	605	885	1135	1354
SZ	2379	2090	2109	2424	2185	1188	605	885	1135	1354
EV	2400	2211	2172	2505	2297	1356	688	1019	1236	1483
PD	2353	2048	2051	2424	2169	1160	585	842	1098	1312

A súlyozott esethez hasonlóan a mohó algoritmus adta a legjobb eredményeket. A legrosszabb pedig ismét az LP relaxáltból kapott kerekítés. Észrevehetjük, hogy a javított mohó és a szintezéses algoritmus majdnem mindig ugyanazt az eredményt adta. A primál-duál algoritmus egy kicsit jobb ezeknél, a legrosszabbnak pedig az élválasztós algoritmus tűnik.

Nézzük, hogy most milyen módon lehet javítani az eredményeket. Olyan pontot kell keresni, amelynek súlya nagyobb, mint a szomszédainak a súlya összesen. Ha egy ilyen pont szomszédait beválasztjuk a lefogásba, akkor biztosan nem növeljük feleslegesen a lefogó ponthalmaz súlyát.

Hivatkozások

[1] Bar-Yehuda, R.; Even, S. *A linear-time approximation algorithm for the weighted vertex cover problem*, Journal of Algorithms **2**, 1981, 2, 198-203

[2] Bar-Yehuda, R.; Even, S. *A local-ratio theorem for approximating the weighted vertex cover problem*, Annals of Discrete Mathematics **25**, 1985, 27-46

[3] Buss, J. F.; Goldsmith, J. *Nondeterminism within P*, SIAM Journal on Computing **22**, 1993, 560-572

[4] Chen, Jianer; Kanj, Iyad A.; Jia Weijia *Vertex cover: further observations and further improvements*, Journal of Algorithms **41**, 2001, 2, 280-301

[5] Chen, Jianer; Liu, Lihua; Jia, Weijia *Improvement on vertex cover for low-degree graphs*, Networks **35**, 2000, 4, 253-259

[6] Chvátal, V. *A greedy heuristic for the set-covering problem*, Mathematics of Operations Research **4**, 1979, 3, 233-235

[7] Clarkson, Kenneth L. *A modification of the greedy algorithm for vertex cover*, Information Processing Letters **16**, 1983, 1, 23-24

[8] Gonzalez, Teofilo F. *A simple LP-free approximation algorithm for the minimum weight vertex cover problem*, Information Processing Letters **54**, 1995, 3, 129-131

[9] Håstad, J. *Some optimal inapproximability results*, J. ACM **48**, 2001, 798-859

[10] Hochbaum, Dorit S. *Approximation algorithms for NP-hard problems*, PWS Publishing Company, Boston, 1997

[11] Khuller, Samir; Vishkin, Uzi; Young, Neal *A primal-dual parallel approximation technique applied to weighted set and vertex cover*, Journal of Algorithms **17**, 1994, 2, 280-289

[12] Nagamochi, Hiroshi; Ibaraki, Toshihide *An approximation of the minimum vertex cover in a graph*, Japan J. Indust. Appl. Math. **16** 1999, 3, 369-375

[13] Savage, Carla *Depth-first search and the vertex cover problem*, Information Processing Letters **14**, 1982, 5, 233-235

[14] Vazirani, Vijay V. *Approximation algorithms*, Springer, 2001