Eötvös Loránd Tudományegyetem

Természettudományi Kar

Fehér Dániel

Független fák keresése hálózatokban

BSc Szakdolgozat

Témavezetők:

Kovács Erika Renáta

ELTE, Operációkutatási Tanszék

Dr. Tapolcai János PhD

BME, Távközlési és Médiainformatiaki Tanszék

Budapest, 2014

Köszönetnyilvánítás

Szeretném megköszönni Kovács Erika Renátának a rengeteg odafigyelést és segítséget. Köszönet Tapolcai Jánosnak a témaválasztásért és az útmutatásért. Köszönet szüleimnek és barátaimnak, akik mindig mellettem álltak.

Tartalomjegyzék

Bevezetés

A szakdolgozat a kommunikációs hálózatokban felmerülő kérdésekkel foglalkozik. Egy kommunikációs hálózat megfeleltethető egy gráfnak, amelyben a pontoknak megfeleltetett objektumok tudnak egymással kommunikálni, és az a két pont van összekötve, amelyek közvetlenül kapcsolatba tudnak egymással kerülni. Feltehető, hogy nincs minden pont közt él, azaz nem tud minden objektum közvetlenül kapcsolatba lépni bármely másik objektummal, de a gráf összefüggő.

Információ továbbítására egy gyakori módszer, amelyet hop-by-hop routingnak is neveznek, mikor minden v pontra rögzítve van egy legrövidebb utak F_v fája, és az információ ezek mentén továbbítódik. Minden csúcs tárolja minden más pontra, hogy abba a pontba melyik élen kell továbbhaladni. Ez azért hasznos, mert így ha egy s pont egy csomagot küld egy másik t pontnak, akkor minden pontnak csak a címzett ismeretére van szükség ahhoz, hogy eldöntse, melyik élen küldje tovább a csomagot.

Ezzel a módszerrel az lehet a probléma, ha például elromlik egy él vagy egy pont, amely benne van valamelyik pont legrövidebb utak fájában. Erre szeretnénk hibakezelést adni úgy, hogy egy ponthoz több feszítőfát is tárolunk. Ez viszont már gráfelméleti probléma, melyet több irányból is megközelíthetünk. Egyrészt két feszítőfa (F_v^1 és F_v^2) esetén ez azt jelenti, hogy minden $u \neq v$ pontra az egyes és a kettes fában lévő u-v utak él- vagy pontdiszjunktak. Az ilyen tulajdonságú fapárokat v gyökerű (él)független fapárnak nevezzük. Egy másik megközelítés, ha nem feltétlenül csak kettő, hanem egy konstans k darab fát tárolhatunk, viszont nem kell feszítőfáknak lenniük. Ilyenkor azt követeljük meg, hogy összesen legyen két fa, amelyben van két különböző diszjunkt útvonal. Ezeket a fákat k-redundáns fáknak nevezzük.

Az s-független fák használatához a routerek például technikailag ma is készen állnak, azonban nincsenek implementálva, mert a javítóutak hosszai sokszorosai is lehetnek az eredeti útvonal hosszának, ahol hossz alatt az út élein lévő élköltségek összegét értjük. A dolgozatban azt vizsgáljuk, hogy a legrövidebb éldiszjunkt útpárokat le tudjuk-e konstans számú (k) fával fedni, ám a dolgozatban belátjuk, hogy ez semmilyen k-ra nem lehetséges. Ismertetünk egy algoritmust, mellyel egy v pont független fapárjait egy fülfelbontás alapján

lineáris időben konstruálhatjuk meg. Azt is vizsgáltuk, hogy egy s-független fapárban az utak összhossza mennyire közelítheti a legrövidebb útpárok értékét. Ezt a feladatot felírtuk egészértékű programozási feladatként is.

Suurballe algoritmusát is részletesen ismertetjük, amely egy pontból az összes többi pontba megtalálja a legrövidebb éldiszjunkt útpárokat. Magától adódik, hogy a Suurballe-algoritmust fel lehet-e használni egy egyszerű heurisztika megadására, és ha igen, akkor abból mennyire lesz jó közelítés. Két ilyen heurisztikát, és magát a Suurballe algoritmust is implementáltuk, és az összehasonlításra kapott teszteredményeket illusztráltuk.

1. fejezet

Független és redundáns fák

1.1. Független fák

1.1.1. Definíció. *Legyen T_1 és T_2 két feszítőfa egy G egyszerű gráfban, és s a G egy csúcsa. T_1 és T_2 s-(él)függetlenek, ha G minden s-től különböző v csúcsa esetén a v-ből s-be T_1-ben illetve T_2-ben vezető egyértelmű utak belsőleg (él)diszjunktak, vagyis nincs sem közös élük, sem pedig közös belső csúcsuk. Beszélhetünk még s-élfüggetlen feszítőfákról is, ekkor a belső utaknak éldiszjunktaknak kell lenniük.*

Általánosíthatjuk a definíciót nem csak kettő, hanem tetszőlegesen sok k-ra is, de ebben a dolgozatban részletesebben a kettő független fa esetét vizsgáljuk. Az ilyen fák keresését még tovább lehet specializálni, például ha $c : E \to \mathbb{R}_+$ súlyozásra a két fa súlyának összegére minimálisakat keressük. Egy másik megközelítés az, amikor a két fában a belsőleg diszjunkt utak a legszélesebbek legyenek, ami azt jelenti, hogy ha egy másik úton haladnánk, az új út élei közt lenne egy kisebb élköltségű tag. Alkalmazásban ez is előfordulhat, ha például az internet hálózatában a legnagyobb sávszélességű utat akarjuk megkeresni. A következőkben belátjuk, mik a feltételei 2 független feszítőfa létezésének, ehhez bevezetünk néhány újabb fogalmat.

1.1.2. Definíció. *Egy $G = (V, E)$ gráf fülfelbontása egy $(P_0, P_1, ..., P_k)$ sorozat, ahol P_0 G-beli kör, P_i, $1 \leq i$ pedig egy G-beli út, amire $V(P_i) \cap (V(P_0) \cup V(P_1) \cup ... \cup V(P_{i-1}))$ a P_i két végpontja és $G = P_0 \cup P_1 \cup ... \cup P_k$. Kétféle fülfelbontás létezik, a nyílt és az általánosított fülfelbontás. A nyíltnál megköveteljük, hogy a bevett P_i út két végpontja egymástól különböző legyen, míg az általánosítottnál ezek a pontok megegyezhetnek.*

A fülfelbontás fontosságát az alábbi közismert tétel mutatja:

1.1.3. Tétel. *Egy gráf akkor és csak akkor 2-(él)összefüggő, ha van nyílt (általánosított) fülfelbontása. Sőt, a fülfelbontásban P_0 a G tetszőleges köre lehet, és általában, ha G egy részgráfját már előállítottuk, az kiegészíthető a G egy fülfelbontásává.* □

1.1.4. Definíció. *Legyen $G = (V, E)$ egy 2-összefüggő gráf és legyen $n = |V|$. Azt mondjuk, hogy egy $g : V \to \{1, ..., n\}$ egy tetszőleges $(s, t) \in E$ élre egy $s - t$-számozás, ha $g(s) = 1$, $g(t) = n$ és minden $v \in V - \{s, t\}$ csúcsnak van u, w szomszédja, hogy $g(u) < g(v)$ és $g(w) > g(v)$.*

Ezt megfogalmazhatjuk a rendezések nyelvén is:

1.1.5. Definíció. *Legyen $G = (V, E)$ egy 2-összefüggő gráf. Azt mondjuk, hogy adott egy $s - t$-rendezés a G gráf csúcsain egy $\prec$ rendezéssel, ha s maximális és t minimális, azaz minden $v \neq s, t$ esetén $s \prec v \prec t$, és minden $v \in V(G) - \{s, t\}$ csúcsnak vannak olyan u, w szomszédjai, amikre $u \prec v \prec w$.*

A két fogalom ekvivalens. Egy nyílt fülfelbontás segítségével könnyen megkaphatjuk a gráf egy $s - t$-rendezését. Vegyünk ugyanis egy olyan $P_0, ..., P_k$ fülfelbontást, ahol P_0 tartalmazza az (s, t) élt, vagyis $P_0 = (s = v_0, v_1, ..., v_{l-1} = t, s)$. Legyen $v_i \prec v_{i+1}, i = 1, ..., l - 2$. Ezzel P_0-on kaptunk egy $s - t$-rendezést, ha pedig $P_0, ..., P_{i-1}$-n már adott, $P_i = (u_0, ..., u_m)$ (feltehető $u_0 \prec u_m$) és w a korábbi rendezésben u_0-ra következő elem, akkor legyen $u_0 \prec u_1 \prec ... \prec u_{m-1} \prec w$. Ezzel könnyen láthatóan egy $s - t$-rendezést kapunk. Így adódott, hogy

1.1.6. Tétel. *Ha egy gráf kettő-összefüggő, és (s, t) egy tetszőleges éle, akkor létezik a gráfnak $s - t$-rendezése.* □

Az $s - t$-rendezés létezéséből már adódik a két független fa létezése:

1.1.7. Tétel. (Itai és Rodeh) *Legyen G egy 2-összefüggő gráf, és (s, t) a G egy tetszőleges éle. Ekkor G-ben létezik két s-élfüggetlen feszítőfa. [5]*

Bizonyítás. Legyen t az s egy szomszédja, z pedig a t egy s-től különböző szomszédja. Tekintsünk a gráfban egy $\prec$ $s - t$-rendezést. Konstruálunk egy T_1 és egy T_2 s gyökerű feszítőfát úgy, hogy a gráf minden s-től különböző csúcsára megmondjuk, hogy mi a szülője a megfelelő fában. T_1-ben legyen t szülője z, minden más $v \in V - \{s, t\}$ csúcs szülője pedig a v egy tetszőleges olyan u szomszédja, amire $u \prec v$. T_2-ben legyen t szülője s, a többi $v \in V - \{s, t\}$ csúcs szülője pedig a v egy tetszőleges olyan w szomszédja, amire $w \succ v$. Az így kapott feszítőfák s-függetlenek. □

A kettő-összefüggő eset felhasználásával levezetjük a kettő-élösszefüggő esetet:

1.1.8. Tétel. (Itai és Rodeh) *Legyen G egy 2-élösszefüggő gráf, és s a G egy tetszőleges csúcsa. Ekkor G-ben létezik két s-független feszítőfa. [5]*

Bizonyítás. Ha G 2-összefüggő, akkor persze készen vagyunk, mert az s-független fák s-élfüggetlenek is. Ha nem, akkor legyenek $B_1, ..., B_m$ a G blokkjai (maximális kettő-összefüggő részgráfjai) úgy, hogy $s \in B_1$. Mivel G kettő-élösszefüggő, a blokkfelbontásban nincsenek elvágóélek, csak blokkok és elvágó csúcsok, és B_1-t a blokk-fa gyökerének tekintve minden B_i-nek ($i > 1$) létezik egy egyértelmű B_j szülője, amihez egy s_i elvágó csúccsal csatlakozik. Legyen $s_1 := s$ és minden B_i ($i = 1, ..., m$) blokkban legyen S_i és T_i két s_i-független feszítőfa (ilyenek léteznek a korábbi tétel szerint), végül legyen $S = \bigcup_{i=1}^{m} S_i$ és $T = \bigcup_{i=1}^{m} T_i$. Ekkor könnyen látható, hogy S és T s-élfüggetlen feszítőfák G-ben. □

1.2. Redundáns fák

Hálózatok hibakezelésénél egy másik hasonló megközelítés a redundáns fák fogalomköre.

1.2.1. Definíció. *Legyen $T_1, T_2, ..., T_k$ k fa egy G egyszerű irányítatlan gráfban, és s a G egy csúcsa. $T_1, T_2, ..., T_k$ s gyökerű k-(pont)redundáns fák, ha bármelyik élet (pontot, ami nem az s) kivesszük a gráfból, továbbra is lesz út bármely pontból az s-be legalább az egyik fában.*

$k = 2$ eset, amelyet csak (pont)redundáns fapárnak nevezünk:

1.2.2. Állítás. *Egy G egyszerű irányítatlan gráfban az élfüggetlen feszítőfapár és a redundáns fapár fogalma ekvivalens.*

Bizonyítás. $\Rightarrow$ Ha T_1 és T_2 élfüggetlen, akkor tetszőleges v pontra az $s-v$- útpár belsőleg éldiszjunkt, tehát ha eltávolítok egy élt a gráfból, akkor valamelyik út továbbra is létezni fog.

$\Leftarrow$ Mivel csak két redundáns fánk van, mindkettőnek feszítőfának is kell lennie, különben lenne olyan pont, amit az egyik fa nem fedne le, és akkor az ahhoz tartozó él törlésével nem teljesülne a tulajdonság. Másrészt, ha lenne olyan útpár a két fában, amelyeknek van közös éle, akkor annak az élnek a törlésével nem teljesülne a redundáns fák tulajdonsága. □

Az állítás ugyanúgy igaz a pontredundáns és pontfüggetlen esetre is, a bizonyítás pedig teljesen hasonló módon működik.

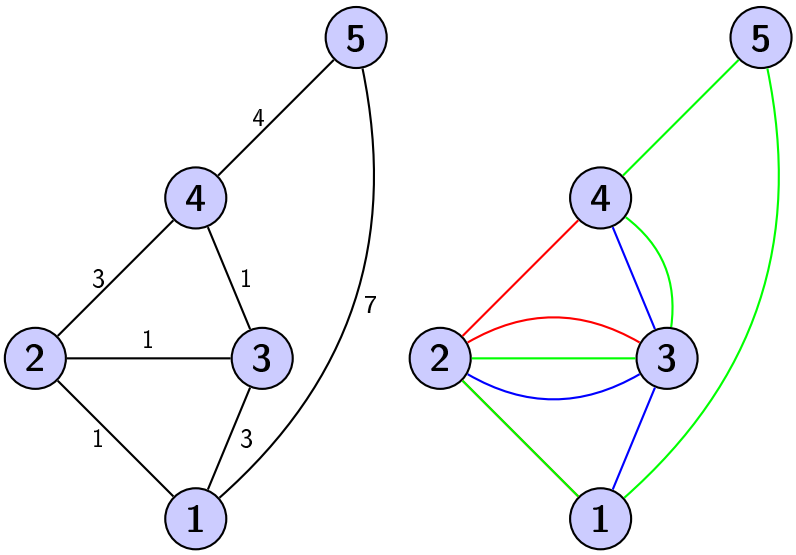
1.3. k-redundáns fák útjaira vett minimalitás

Nézzük azt az esetet, amikor az s és az összes többi pont között adottak minimális súlyú útpárok. Az nem feltétlenül teljesül, hogy ezeket az utakat be lehet rendezni egy független fapárba, de meg kell vizsgálnunk, hogy vajon k-redundáns fákkal le lehet e fedni őket, ahol k véges.

1.3.1. Állítás. *Minden $k \in \mathbb{N}$-re létezik G gráf, hogy az s-ből induló legrövidebb útpárok nem fedhetőek le k-redundáns fákkal.*

Bizonyítás. Minden k-ra tudunk generálni egy G_k gráfot, amire igaz lesz az állítás.

k = 2 eset G_2 legyen az alábbi gráf (az 1-es csúcs a gyökér):

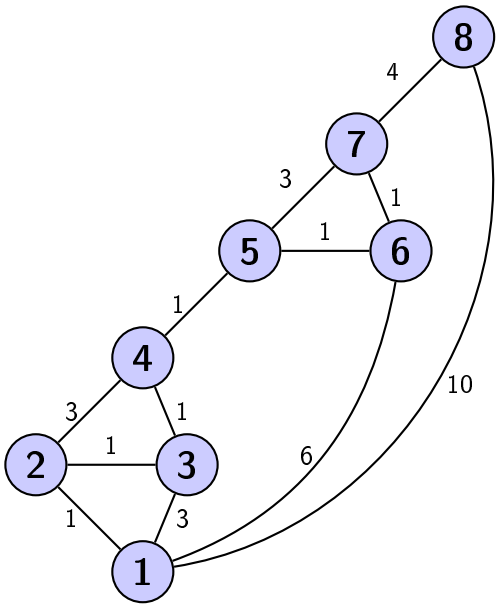


1.1. ábra. $k = 2$ eset

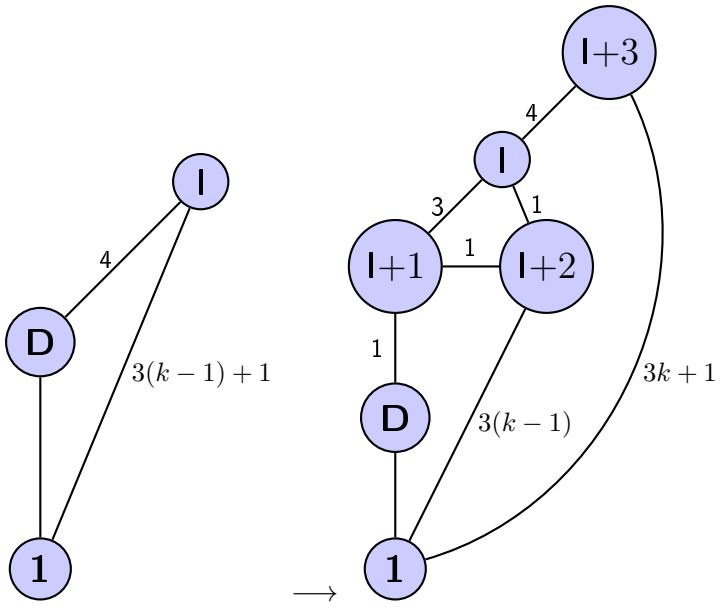
Látható, hogy az 5-ös pontból érkező útvonalat nem lehet sem a piros, sem a kék fába belerakni, tehát 2 fával nem fedhetők le az útvonalak.

k = 3 eset G_3 legyen az 1.2 ábra, továbbra is az 1-es pont a gyökér. Itt is könnyen látható, hogy az 5-ös pont útvonalai miatt nem elég 2 fa, viszont a 8-as pont útvonalai, mivel az 5-6-7 vonalon halad az egyik oldalon, további fákat igényelnek, azaz már 3 fa sem elég.

Tetszőleges k-as eset ($k > 2$) Tetszőleges $k \in \mathbb{N}$ esetre pedig visszavezetjük a $k-1$-es esetre az 1.3 ábra alapján.



1.2. ábra. $k = 3$ eset



1.3. ábra. $(k-1)$-es esetből a k-as képzése

Az ábrában D a már meglévő rész, amelyen nem módosítunk, I pedig G_{k-1} ábra szerinti legfelső pontja. Vegyünk be 3 új pontot a következőképpen. A D ponthalmaz és az I pont közötti 4 súlyú élt osszuk fel egy $I+1$-edik ponttal, úgy, hogy az $(I+1, I)$ él súlya 3, a D ponthalmaz és az $I+1$-edik csúcs közt él hossza pedig 1 legyen. Az

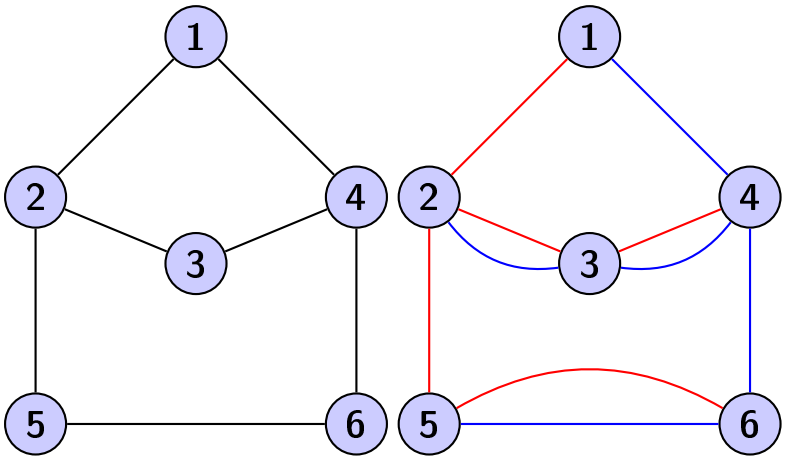
$(1, I)$ $3k+1$ súlyú élt is felosztjuk egy $I+2$-edik ponttal, ahol az $(1, I+2)$ él súlya $3k$, míg az $(I+2, I)$ él súlya 1 legyen, valamint adjunk még a gráfhoz egy 1 súlyú $(I+1, I+2)$ élt. Harmadiknak vegyünk a gráfhoz egy $I+3$ pontot, amelyhez két él csatlakozik, egy $(I, I+3)$ él 4 súllyal, és egy $(1, I+3)$ él $3k+1$ súllyal. Az így kapott gráf legyen G_k.

□

1.4. Algoritmus független fák megadására

Ha már adott egy $s-t$-rendezésünk, akkor a független fák létrehozása nem túl nehéz feladat. Legyen T_R és T_B a két fa, és az alapkör feldolgozásánál a kör összes élét, kivéve az (s,t)-t hozzáadjuk T_R-hez, míg az (s, v_1) él kivételével az összeset a T_B-hez.

Egy tetszőleges fül feldolgozásánál attól függően, hogy a $P_i = \{v_k, v_{k+1}, ..., v_l\}$ fül végpontjai milyen relációban állnak egymással, egy-egy kivételével hozzáadjuk a már meglévő fákhoz az összes élt, mégpedigy úgy, hogy ha $v_k \prec v_l$, akkor $T_R = T_R \cup \{(v_l, v_{l-1}), (v_{l-1}, v_{l-2}), ..., (v_{k+2}, v_{k+1})\}$, $T_B = T_B \cup \{(v_k, v_{k+1}, (v_{k+1}, v_{k+2}), ..., (v_{l-2}, v_{l-1})\}$. Ellenkező esetben mindkét élhalmazt éppen a másik fához csatoljuk hozzá. [3]



1.4. ábra. Egy egyszerű független fapár fülfelbontásból

A fák függetlensége az $s-t$-rendezésből azonnal következik, hiszen a két fában minden pontra éppen a rendezés szerinti két útvonal szerepel.

procedure $s-t$-RENDEZÉS$(G, s, P_0, ..., P_k)$
$P_0 = \{v_0, v_2, ..., v_j\}$, $v_0 := s$

for $v_i \in P_0, i = 1, ..., j$ **do**

$v_i \prec v_{i-1}$

end for

$T_R := \{(v_0, v_1), (v_1, v_2), ..., (v_{j-1}, v_j)\}$, $T_B := \{(v_0, v_j), (v_j, v_{j-1}), ..., (v_2, v_1)\}$

for $i = 1, ..., k$ **do**

$P_i = \{v_l, ..., v_j\}$

if $v_j \prec v_l$ **then**

for $v_t \in P_i, t = l + 1, ..., j - 1$ **do**

$v_t \prec v_{t-1}$

end for

$T_R := T_R \cup \{(v_l, v_{l-1}), (v_{l-1}, v_{l-2}), ..., (v_{j+2}, v_{j+1})\}, T_B := T_B \cup \{(v_k, v_{j+1}), (v_{j+1}, v_{j+2}), ..., (v_{l-}$

else

for $v_t \in P_i, t = j - 1, ..., t + 1$ **do**

$v_t \prec v_{t+1}$

end for

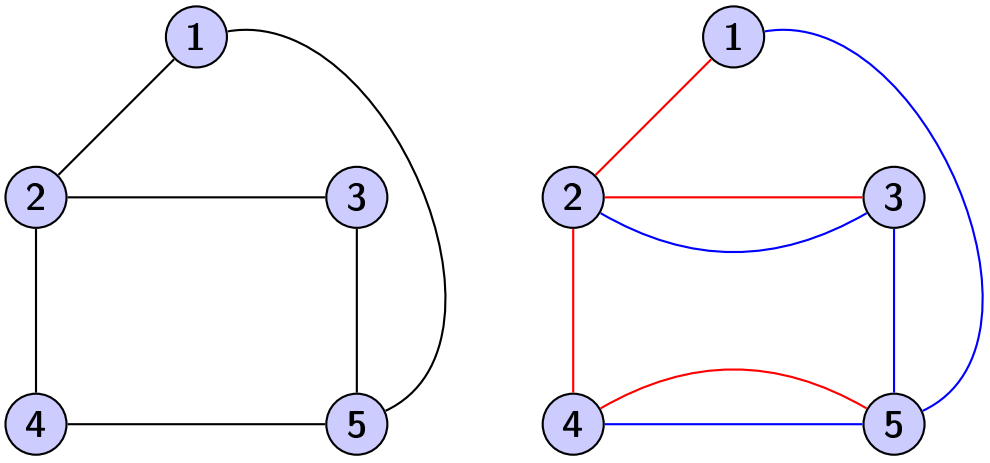
$T_B := T_B \cup \{(v_l, v_{l-1}), (v_{l-1}, v_{l-2}), ..., (v_{k+2}, v_{k+1})\}, T_R := T_R \cup \{(v_k, v_{k+1}), (v_{k+1}, v_{k+2}), ..., (v_{l-}$

end if

end for

end procedure

1.4.1. Állítás. *Nem minden független fapár áll elő fülfelbontásból.*



1.5. ábra. Példa arra, hogy nem minden független fapár áll elő fülfelbontásból

A fenti példában bármely pont lehet a gyökér, de egyikre sem lehet alapkört meghatározni a két fa alapján.

2. fejezet

Suurballe algoritmusai

A minimális súlyú javítóutak témájához szorosan kapcsolódnak J. W. Suurballe algoritmusai [1, 2], amelyekkel a következő problémát tudjuk megoldani: adott $G = (V, E)$ erősen összefüggő irányított gráf nemnegatív c élsúlyokkal és $s \in V$. Keresünk minden $v \in V - s$ pontra s-ből egy P és egy Q éldiszjunkt útpárt, melyre $c(P) + c(Q)$ minimális.

2.1. Suurballe algoritmusa két pont közti éldiszjunkt legrövidebb útpárra

Legyen $G = (V, E)$ egy irányított gráf, $|V| = n$ darab csúccsal, amelyek közül s legyen a kiemelt forrás, $|E| = m$ darab éllel, $c : E \to \mathbb{R}$ nemnegatív élsúlyozással, és G legyen antiszimmetrikus, azaz ha $(v, w) \in E$, akkor $(w, v) \notin E$. Futtassunk egy Dijkstra-algoritmust az s-ből, az így kapott legrövidebb utak fenyője legyen T, és minden $v \in V$ pont távolsága s-től legyen $d(v)$, $0 \leq d(v)$, $d(s) = 0$. Megadunk egy c' módosított élköltséget minden $(v, w) \in E$ élre, mégpedig úgy, hogy $c'(v, w) = c(v, w) - d(w) + d(v)$. Az így kapott gráfunkra igaz az alábbi tulajdonság, amely a minimális költségű folyamok tételéből következik:

2.1.1. Állítás. *Minden $(v, w) \in E$ élre $c'(v, w) \geq 0$, és ha $(v, w) \in T$, akkor egyenlőség áll fent.*

Ha p egy tetszőleges út egy v pontból egy w pontba, akkor $c'(p) = c(p) - d(w) + d(v)$. Így két utat ugyanazzal a kezdő és végponttal ugyanannyival változtattunk meg, és az ilyen utak rendezése nem változik a transzformációval. Így a feladatot elég erre az élsúlyozásra megoldani, és ebből vissza tudjuk kapni az eredeti költségű utakat.

Válasszunk ki egy v pontot, amelybe szeretnénk megkeresni a páronként éldiszjunkt legrövidebb útpárt. Erre a pontra definiáljunk egy G_v gráfot, ami legyen az, hogy G-ben megfordítjuk a T-ben levő $s - v$ útbeli éleket, a c' szerinti költségük pedig marad 0. Az így

kapott G_v gráfban futtassunk egy Dijkstra-algoritmust, és a kapott költségeket jelöljük egy tetszőleges $w \in V$ $d_v(w)$-vel. Ennek a távolságnak a v-re vonatkozó értékéről szól az alábbi tétel, amely azonnal következik a minimális költségű folyamokról szóló Ford-Fulkerson-tételből:

2.1.2. Tétel. *A $d_v(v)$ érték pontosan megegyezik a páronként éldiszjunkt minimális súlyú $s-v$ útpárok c' szerinti összhosszával. Az utakat úgy kaphatjuk meg, hogy vesszük a T és a G_v-beli legrövidebb $s-v$ utak unióját, és ebből kivesszük az összes olyan élt, amelynek a fordítottja is benne van, a maradék éleket pedig szétosztjuk két úttá.*

procedure SUURBALLE $1(G, c, s, v; P)$ ▷ A G gráfra hívjuk meg c élsúlyozással, az s-ből a v-be keressük az útpárt, amelynek éleit P-ben adjuk vissza

$Dijsktra(G, s, c; T, d)$ ▷ meghívunk egy Dijsktra-algoritmust a G gráfra s gyökérrel, c élsúlyozással, amely a T legrövidebb utak fáját, és $\forall\ v \in V$ pontokra a $d(v)$ értékeket adja vissza

for $(v, w) \in E$ **do**

$c'(v, w) := c(v, w) - d(w) + d(v)$

end for

$G_v := G, x := v$

while $x \neq s$ **do**

$E(G_v) := E(G_v) - (p_T(x), x) + (x, p_T(x))$ ▷ ahol $p_T(x)$ az x pont szülője a T fában

$x := p_T(x)$

end while

$Dijsktra(G_v, s, c'; T_v, d_v)$

$x := v$

while $x \neq s$ **do**

$P := P \cup (p_T(x), x)$

$x := p_T(x)$

end while

$x := v$

while $x \neq s$ **do**

if $(x, p_{T_v}(x)) \in P$ **then**

$P := P - (x, p_{T_v}(x))$

else

$P := P \cup (p_{T_v}(x), x)$

end if

$x := p_{T_v}(x)$
end while
end procedure

2.2. Suurballe algoritmusa az általános esetre

Általános eset alatt azt értjük, amikor egy s csúcsból minden $v \neq s$ pontba keressük az éldiszjunkt legrövidebb útpárokat. A G gráfunkat definiáljuk úgy, mint előbb, és ugyanúgy készítsük el az s-ből induló legrövidebb utak költségeivel a c' súlyozást. Az algoritmus minden pontra tárolja a következő értékeket:

- Egy L halmaz, amelyben a megcímkézett pontokat tároljuk
- $d : V \to \mathbb{R}$ nemnegatív súlyozás, amelyet úgy inicializálunk, hogy $d(s) = 0$, és minden $v \in V - s$ pontra $d(v) = \infty$
- $p, q : V \to V$, amelyek a kezdetben nincsenek meghatározva semelyik pontra sem

Még egy további dolgot jegyzünk fel, mert amikor eltávolítunk egy megcímkézett pontot a legrövidebb utak fájából, azaz T-ből, akkor az szétesik címkézetlen pontok részfáira. Az algoritmus számon tartja ezeket a címkézetlen részfákat. Kezdésként csak egy címkézetlen részfa van, ami maga a T fa. Az algoritmus a következő lépésből áll, és addig fut, amíg van olyan címkézetlen v pont, hogy $d(v) \neq \infty$.

Válasszuk ki a címkézetlen csúcsok közül azt, amelyiknek a $d(v)$-je minimális. Legyen S a v-t tartalmazó címkézetlen részfa. Címkézzük meg $v - t$, ez felosztja S-t új címkézetlen részfákká. Minden nem fa (u, w) élre, amely olyan, hogy u és w is S-ben van és vagy $u = v$, vagy u és w különböző címkézetlen részfákban vannak v megcímkézése után, ha $d(v) + c'(u, w) < d(w)$, akkor $d(w) := d(v) + c'(u, w)$, $p(w) := u$, $q(w) := v$.

2.2.1. Tétel. *Miután az algoritmus lefutott, $d(v)$ megegyezik a páronként éldiszjunkt minimális súlyú $s - v$ útpár értékével a c' súlyozás mellett (ha $d(v) = \infty$, akkor nem létezik ilyen útpár). Ha $v \neq s$ és $d(v) < \infty$, akkor az előző algoritmusban meghatározott G_v gráfban létezik egy út s-ből v-be, amelynek a súlya $d(v)$ és az utolsó éle $(p(v), v)$.*

Bizonyítás. Legyen v tetszőleges pont. G_v legyen a két pont közti legrövidebb éldiszjunkt útpárra vonatkozó Suurballe-algoritmusban ismertett G_v gráf, és ezen futtatjuk a Dijsktra-algoritmust. Ezt úgy tudjuk megtenni, hogy fent tartjuk a következő invariáns tulajdonságot, ahol d_v és p_v jelölje folyamatában a Dijsktra-algoritmus szerinti távolság és szülő függvényeket.

Egy x csúcs akkor és csak akkor címkézetlen G_v-ben, ha G-ben ugyanabban a címkézetlen pontok részfájában van, mint v. Ha x címkézetlen G_v-ben, $d_v(x) = d(x)$ és $p_v(x) = p(x)$ (ha mindkettő definiálva van). Ha x címkézett G_v-ben, akkor $d_v(x) = d(y)$, ahol y az a csúcs, amelynek G-beli megcímkézésével x-et eltávolítottuk a v-t tartalmazó címkézetlen pontok részfájából. Ha $x = y$, akkor $p_v(x) = p(y)$.

Ez az invariáns tulajdonság az inicializálás után teljesülni fog. Tegyük fel, hogy a tulajdonság megmarad az előtt, hogy a legrövidebb útpár algoritmusunk megcímkéz egy y pontot G-ben. Ekkor két eset lehetséges:

1. eset (y címkézett G_v-ben) Az invariáns tulajdonság miatt y és v különböző címkézetlen pontok részfáiban vannak közvetlenül y megcímkézése előtt. y címkézése semelyik változót nem változtatja meg a v-t tartalmazó címkézetlen részfában és így nem befolyásolja az invariáns tulajdonságot.Az y pont címkézésének szimulálásánál G-ben a Dijsktra-algoritmus futtatása G_v-n nem változtat semmit.

2. eset (y címkézetlen G_v-ben) Az invariáns tulajdonság miatt y és v ugyanabban a címkézetlen pontok részfájában vannak közvetlenül y megcímkézése előtt. Legyen S ennek a részfának a pontjainak a halmaza, S' pedig azon pontok halmaza, amelyek a v-t tartalmazó címkézetlen pontok részfájában vannak y megcímkézése után (ha $y = v$, akkor S' üres lesz). Az invariáns tulajdonság miatt $d_v(y)$-nak minimálisnak kell lennie a G_v-ben levő címkézetlen pontok között. Hogy szimuláljuk y megcímkézését G-ben, a Dijsktra-algoritmus G_v-n futtatva először megcímkézi y-t, majd a maradék pontokat az $S - S'$ halmazban egy megfelelő sorrendben, mivel minden pont elérhető G_v-ben y-ból egy néhány 0 súlyú visszaélből és néhány 0 súlyú faélből álló úton. Könnyen látható, hogy ez a szimuláció megtartja az invariáns tulajdonságot.

Így az invariáns tulajdonság mindig megmarad. □

procedure SUURBALLE 2$(G, c, s; p, q)$ ▷ A G gráfra hívjuk meg c élsúlyozással, s a gyökér, visszaadja minden pont p és q értékét

```
Dijsktra(G, s, c; T, b)
for (v, w) ∈ E do
    c'(v, w) := c(v, w) − b(w) + b(v)
end for
for v ∈ V do
    if v = s then
        d(v) := 0
    else
```

$d(v) := \infty$

end if

end for

$L := \emptyset$, $\bar{L} := V$

while $\exists\ v \in \bar{L}$, $d(v) \neq \infty$ **do**

$v := argMin(\bar{L})$ ▷ az argMin függvény kiválasztja a legkisebb d-jű címkézetlen csúcsot

$L := L \cup \{v\}$, $\bar{L} := \bar{L} - \{v\}$

$S = S_1 \cup ... \cup S_i \cup v$ ▷ ahol S a v-t tartalmazó címkézetlen pontok részfája, és az S_j-k az $S - v$ címkézetlen pontjainak részfái

for $u, w \in S$, $(u, w) \notin S$, $u \in S_j \Rightarrow w \notin S_j$, $w \neq v$ **do**

if $d(v) + c'(u, w) < d(w)$ **then**

$d(w) := d(v) + c'(u, w)$, $p(w) := u$, $q(w) := v$

end if

end for

end while

end procedure

Így legeneráltuk minden pontra a $p(v)$, $q(v)$ értékeket, amelyekből a következő módon visszakaphatjuk egy tetszőleges v és az s közötti útpárt.

Minden pontnak legyen egy logikai tulajdonsága, mégpedig az, hogy megjelöltük-e, vagy sem. Kezdésként legyen minden pont jelöletlen, legyen egy x változónk, és $x := v$. Jelöljük meg x-et, és utána $x := q(x)$, amíg $x \neq s$. Ennek muszáj s-ben végződnie, hiszen minden x pontra $q(x)$-et hamarabb címkéztük meg, és s-et címkéztük meg először. Ezután a következő algoritmust kétszer elvégezzük, amivel visszakapjuk a két út éleit.

Legyen $x := v$, és addig iterálunk, míg $x \neq s$. Ha x jelölt, tegyük jelöletlenné, adjuk hozzá az úthoz a $(p(x), x)$ élt, és $x := p(x)$. Ha x nem jelölt, akkor adjuk hozzá az úthoz a $(p_T(x), x)$ élt, és $x := p_T(x)$, ahol $p_T(x)$ az x pont őse az s gyökerű legrövidebb utak fájában.

procedure SUURBALLE ÚTGENERÁLÁS$(G, v, p, q, T; P_1, P_2)$ ▷ A v pontba akarjuk konstruálni a két utat, amelyeket a P_1 és P_2 élsorozatokban adunk vissza, a T pedig az s gyökerű legrövidebb utak fája

$M := \emptyset$, $\bar{M} := V$, $x := v$ ▷ Az M a jelölt, míg az $\bar{M}$ a jelöletlen pontok halmaza

while $x \neq s$ **do**

$M := M \cup \{x\}$, $\bar{M} := \bar{M} - \{x\}$

$x := q(x)$

```
    end while
    x := v
    while x ≠ s do
        if x ∈ M then
            P_1 := P_1 ∪ (p(x), x), M := M − {x}, M̄ := M̄ ∪ {x}
            x := p(x)
        else
            P_1 := P_1 ∪ (p_T(x), x)
            x := p_T(x)
        end if
    end while
    x := v
    while x ≠ s do
        if x ∈ M then
            P_2 := P_2 ∪ (p(x), x), M := M − {x}, M̄ := M̄ ∪ {x}
            x := p(x)
        else
            P_2 := P_2 ∪ (p_T(x), x)
            x := p_T(x)
        end if
    end while
end procedure
```

2.2.1. Az algoritmus implementálása

A címkézetlen pontok tárolására ugyanúgy használhatunk d-kupacot, ahol $d = \theta(1+m/n)$, mint a Dijsktra-algoritmusnál, ez egyetlen adódó komplikáció az az, hogy szükségünk van egy módszerre, hogy meghatározzuk, mely éleket dolgozzuk fel. Egy (u, w) élet feldolgozottnak nevezünk, ha u és w ugyanabban a részfában volt v megcímkézése előtt és vagy $u = v$, vagy u és w különböző címkézetlen pontok részfáiban vannak v megcímkézése után, így v megcímkézésével leellenőrizzük a $d(v) + c(u, w) < d(w)$ egyenlőtlenséget.

Az ilyen élek megadásához egy három részből álló adatstruktúrát adunk meg. Az első rész a T fa pontjainak preorder és posztorder számozása. A v pont preorder számának jele legyen $pre(v)$, posztorder számának pedig $post(v)$. Ezek a számozások egyszerűvé teszik az ős-leszármazott kapcsolat ellenőrzését: a v csúcs a w őse, ha $pre(v) \leq pre(w)$ és $post(v) \geq post(w)$.

A második rész reprezentálja a címkézetlen pontok rendezését a címkézetlen pontok

részfáiba. Minden címkézetlen v pontra tárolunk egy $children(v)$ kétirányú láncolt listát, amelyben a v címkézetlen gyerekeit tároljunk. Továbbá tároljuk még a v pont szülőjét a T fában $p(v)$ néven, ha az létezik és címkézetlen, különben $p(v) := null$.

A harmadik rész egy $attaching(v)$ kétirányú láncolt lista, amely a v-be belépő vagy v-ből kilépő feldolgozatlan nem fa éleket tartalmazza. Egy ilyen listában rendezzük az éleket monoton növő sorban az él másik végpontja szerinti preorder számok szerint.

Egy v pont megcímkézésénél a következőképpen frissítjük a címkézetlen pontok részfáit és a feldolgozandó éleket. Először megvizsgáljuk az $attached(v)$ listát. Minden (v, w) és (w, v) élre a listából, töröljük az élt az $attached(v)$ és az $attached(w)$ listákból, és ha az él a v-ből lép ki, azt az élt feldolgozzuk.

A következő lépés a létrejövő részfák pontjainak megvizsgálása. Ez másképp működik arra a részfára, amely a $p(v)$ pontot tartalmazza, mint a v gyerekei által meghatározott részfákra. Ha $p(v) \neq null$, akkor a $p(v)$-t tartalmazó részfa minden x pontját egyesével végigjárjuk, egy x-re pedig megvizsgáljuk az $attached(x)$ listát. Minden egyes (u, w) élre megnézzük, hogy az él nem x vége v leszármazottja-e. Ha az, akkor töröljük az (u, w) élt az $attached(u)$ és $attached(w)$ listákból, és feldolgozzuk az élt, különben nem csinálunk vele semmit (Ebben ez esetben nevezzük az (u, w) élt elpazaroltnak).

A v egy y gyerekét tartalmazó részfát is pontonként vizsgáljuk meg. A részfa egy x pontjára átnézzük az $attached(x)$ listát az elejéről sorban haladva. Minden (u, w) élre ellenőrizzük, hogy az él nem x vége az y leszármazottja-e. Ha nem, akkor töröljük az (u, w) élt az $attached(u)$ és $attached(w)$ listákból, és feldolgozzuk az élt, különben nem csinálunk vele semmit (ezt az élt elpazaroltnak hívjuk), hanem a lista végéről visszafelé indulva elkezdjük az élek ellenőrzését, és ugyanazt tesszük mint az előbb. Kivételt képez, ha a vizsgált él másik vége az y leszármazottja, mert akkor nem teszünk vele semmit (ezt az élt elpazaroltnak hívjuk), és a részfa következő pontjára lépünk. Az $attached(x)$ lista rendezése biztosítja, hogy a feldolgozandó élek két összefüggő csoportban, a lista elején és a végén szerepeljenek, és így az előről és a hátulról való ellenőrzés minden szükséges élt megtalál.

A különböző címkézetlen pontok részfái megvizsgálását egyszerre hajtjuk végre úgy, hogy egyszerre csak egy lépést teszünk egy részfában, aztán egy lépést egy másik részfában, és így tovább, amíg egy kivételével minden részfát végigvizsgáltunk. Egy lépés egy részfában addig tart, amíg nem találunk egy elpazarolt élt, vagy ha vágeztünk egy pont vizsgálatával. Azért nincs szükségünk az utolsó részfa végigvizsgálására, mert addigra már csak azon a fán belül lehet feldolgozatlan él.

Az egyetlen probléma az lehet, ha törlünk egy olyan élt, amelynek vizsgálatát éppen felfüggesztettük egy másik részfában. Ezt úgy tudjuk megelőzni, hogyha törlünk egy él az

attached(x) listából, meg kell néznünk, van-e ezen az élen álló vizsgálat, és ha igen, akkor az arra az élre mutató pointert eggyel előre vagy hátra állítjuk az *attached*(x)-ben, attól függően, hogy éppen előlről vagy hátulról haladunk végig az éleken.

2.2.2. Tétel. *Ezzel az implementációval az algoritmus $O(m\ log_{1+m/n}n)$ időben fut, a szükséges tárhely pedig $O(m)$ nagyságrendű. [2]*

2.3. Az algoritmus átalakítása általános gráfokra és pontdiszjunkt esetre

Az algoritmus könnyen alkamazható akkor is, ha G nem antiszimmetrikus, hanem megengedünk oda-vissza éleket is, ebbe bele tartoznak az irányítatlan gráfok is, ha egy irányítatlan gráf minden élét oda-vissza megirányítom. Ehhez csak annyit kell tennünk, hogy a $p(v)$ függvényünk a megelőző pont helyett a v-be belépő utolsó élet adja vissza.

A pontdiszjunkt útpárokra való visszavezetés sem nehéz feladat. Egy tetszőleges G antiszimmetrikus irányított gráfra létrehozunk egy G' gráfot, mégpedig úgy, hogy minden egyes v pontot felosztjuk v_1, v_2 pontokra, és összekötjük őket egy (v_1, v_2) 0 súlyú éllel, egy (v, w) élből pedig (v_2, w_1) él lesz. Miután ezzel készen vagyunk, megoldjuk az éldiszjunk útpárok problémáját, ha az eredetiben az s pont volt a gyökér, akkor s_2 gyökérrel.

Egy harmadik általánosítási mód pedig, ha megengedünk negatív súlyú éleket is. Ebből két fajta lehet. Az első, ha nincsen G-ben negatív kör. Ekkor a Dijsktra-algoritmus helyett használjuk a Bellman-Ford algoritmust a legrövidebb utak fájának elkészítéséhez, és ez alapján módosítsuk az élköltségeket. A második, ha van benne negatív kör. Ekkor annak az eldöntése, hogy egy kijelölt s pontból egy választott t-be egy éldiszjunkt útpár a legrövidebb-e, NP-teljes [2].

3. fejezet

Minimális súlyú független fák

Egy fapár súlya alatt azt értjük, hogy minden $v \neq s$ pontra összeadjuk a két fában lévő s-v útvonalak hosszát. Ebben a fejezetben azt vizsgáljuk, egy fapár súlya .

3.1. A probléma modellezése egészértékű programozási feladatként

Megjegyezzük, hogy ha irányítatlan gráffal dolgozunk, akkor egy $[u, v]$ irányítatlan él helyett vegyük az (u, v), (v, u) irányított éleket, és $c[u, v] = c(u, v) = c(v, u)$. Az r pont a gyökér, $\mathcal{T}_1$, $\mathcal{T}_2$ pedig a két fa.

$$\min \sum_{e \in E} \{c_e * \sum_{s \in V \setminus \{r\}} (x_e^{s,\mathcal{T}_1} + x_e^{s,\mathcal{T}_2})\} \tag{3.1}$$

célfüggvénye a következő lineáris programozási feladatnak:

$$\forall s \in V \setminus \{r\}, \mathcal{T}_i \in \{\mathcal{T}_1, \mathcal{T}_2\}, \forall v \in V :$$

$$\sum_{(v,w) \in E} x_{(v,w)}^{s,\mathcal{T}_i} - \sum_{(w,v) \in E} x_{(w,v)}^{s,\mathcal{T}_i} = \begin{cases} -1 & \text{, ha } v = s \\ 1 & \text{, ha } v = r \\ 0 & \text{, különben} \end{cases} \tag{3.2}$$

$$\forall (v, w) \in E, \forall s \in V \setminus \{r\} : x_{(v,w)}^{s,\mathcal{T}_1} + x_{(v,w)}^{s,\mathcal{T}_2} + x_{(w,v)}^{s,\mathcal{T}_1} + x_{(w,v)}^{s,\mathcal{T}_2} \leq 1 \tag{3.3}$$

$$\forall e \in E, \forall s \in V \setminus \{r\}, \mathcal{T}_i \in \{\mathcal{T}_1, \mathcal{T}_2\} : x_e^{s,\mathcal{T}_i} \leq y_e^{\mathcal{T}_i} \tag{3.4}$$

$$\forall v \in V, \mathcal{T}_i \in \{\mathcal{T}_1, \mathcal{T}_2\} : \sum_{(v,w) \in E} y_{(v,w)}^{\mathcal{T}_i} \leq \begin{cases} 1 & \text{, if } v \neq r \\ 0 & \text{, if } v = r \end{cases} \tag{3.5}$$

A (3.1)-es formulában minimalizáljuk a két fa összköltségét. A (3.2)-es egyenlet formalizálja a folyam tulajdonságot minden s nyelő pontra az r gyökérből. A (3.3)-as egyenlőtlenségben

a két fa függetlenségét állítjuk be a folyam változókkal az ellentétes éleken használva. A (3.4)-es egyenlőtlenség meghatározza az indikátor változókat, amelyeket a faélek pozíciójának meghatározására használunk majd. A (3.5)-ös egyenlőtlenségben pedig az indikátor változókon keresztül meghatározzuk a fa tulajdonságot.

3.2. Heurisztikák a feladatra

Kézenfekvő megoldás, ha a Suurballe-algoritmus által megadott útvonalakból próbálunk meg egy heurisztikát létrehozni. Ha tudnánk adni egy fülfelbontást az algoritmus alapján, akkor a korábban ismertetett módszerrel azonnal adódna a két független fánk. Két módszert mutatunk be a fülek meghatározására, melyek abban térnek el, hogy milyen élsúlyozás szerint vesszük a minimális útpárokat.

Az első megközelítés az, ha a redukált élköltségeken vesszük az útpárokat monoton növő sorrendben. Ez azt jelenti, hogy amilyen sorrendben a Suurballe-algoritmus dolgozza fel a pontokat, nekünk is éppen abban a sorrendben kell őket feldolgozni, így ezt bele tudjuk építeni a Suurballe-algoritmusba. Hogy ezt meg tudjuk tenni, a pontoknak legyen még egy logikai tulajdonsága, hogy a pontot lefedtük-e már valamelyik füllel, ennek a neve legyen egy v pontra $covered(v)$. A H_0 alapkört úgy kapjuk meg, hogy vesszük a legelső v pontot, amit a Suurballe-algoritmus feldolgoz. Az ehhez a ponthoz tartozó utak éppen egy kört alkotnak, mivel ha lenne metszéspont a két útvonalban, akkor annak a távolsága kevesebb lenne, mint a v_0-é, és legyen ez a kör a H_0, és a körön lévő összes pont *covered* logikai változóját állítsuk át igazzá.

Ezután vegyük egy tetszőleges v_i pont feldolgozását. Ha a $covered(v_i)$ igaz, akkor nem csinálunk vele semmit. Ha hamis, akkor ehhez a ponthoz is fog tartozni két útvonal az s-ből. Külön-külön nézzük végig az útvonal pontjait a v_i-től kezdve, és hogyha eljutunk egy olyan t pontig, amelyre a $covered(t)$ igaz, akkor megállunk, és ez lesz a H_i fül egyik oldala, majd ugyanezt megtesszük a másik úton is, és így megkapjuk a H_i fül másik felét is. Állítsuk át a H_i fül pontjainak *covered* értékét igazra, és lépjünk a következő pontra, míg fel nem dolgoztuk az összes pontot. Ennek a módszernek az az előnye, hogy lehet egyszerre futtatni a Suurballe-algoritmussal, és így a lépésszámon nem sokat növel, hiszen egy ilyen kör vagy fül megadása $O(m)$ lépésszámban történik, és nem kell minden egyes ponttal együtt meghívni, hanem csak azokra, amelyek még nincsenek lefedve.

procedure SUURBALLE FÜLFEBONTÁS$(G, v, p, q, T, C; H)$ ▷

A v ponthoz akarjuk konstruálni a fület, amelyet a H élsorozatban adunk vissza, C a *covered* igaz elemeit tartalmazza, a T pedig az s gyökerű legrövidebb utak fája

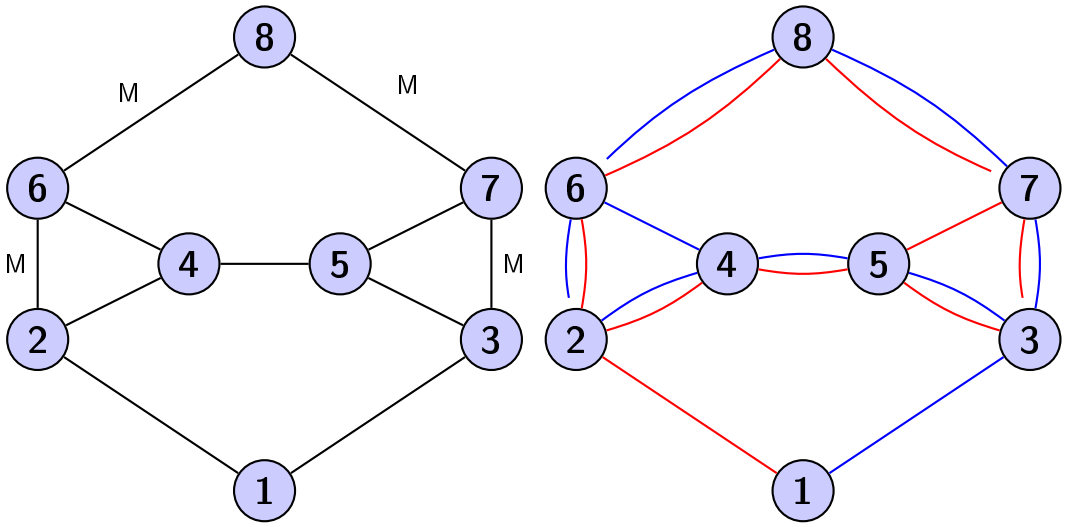
$M := \emptyset$, $\bar{M} := V$, $x := v$ ▷ Az M a jelölt, míg az $\bar{M}$ a jelöletlen pontok halmaza
while $x \neq s$ **do**
$M := M \cup \{x\}$, $\bar{M} := \bar{M} - \{x\}$
$x := q(x)$
end while
$x := v$
while $x \neq s$ & $x \notin C$ **do**
if $x \in M$ **then**
$H := H \cup (p(x), x)$, $M := M - \{x\}$, $\bar{M} := \bar{M} \cup \{x\}$, $C := C \cup x$, $x := p(x)$
else
$H := H \cup (p_T(x), x)$, $C := C \cup x$, $x := p_T(x)$ ▷ Ahol $p_T(x)$ az x őse a T fában
end if
end while
$x := v$
while $x \neq s$ & $x \notin C$ **do**
if $x \in M$ **then**
$H := H \cup (p(x), x)$, $M := M - \{x\}$, $\bar{M} := \bar{M} \cup \{x\}$, $C := C \cup x$, $x := p(x)$
else
$H := H \cup (p_T(x), x)$, $C := C \cup x$, $x := p_T(x)$
end if
end while
end procedure

A második megközelítés annyiban tér el, hogy a valódi élköltség szerinti sorrendben dolgozzuk fel a pontokat. Ehhez először minden pontra meghatározzuk az útpár valódi költségét, majd egy *heapsort* algoritmussal növekvő sorrendbe rendezzük őket. Ezután ugyanazt az algoritmust csináljuk, mint az előbb. Ennek az a hátránya, hogy ki kell számolni az útvonalak hosszát, majd a pontokat sorba kell rendezni.

3.3. Eredmények

3.3.1. Minimális súlyú útpárok és független fák közti hányados

Szintén érdekes kérdés, hogy vajon mekkora lehet a maximális különbség a legrövidebb útpárok értéke és a független fákon vett utak összegére minimális érték között, ha létezik egyáltalán ilyen konstans. Jelenlegi ismereteink alapján a $c \equiv 1$ élsúlyozásra ez a különbség legfeljebb 5/3 lehet, amelyet az alábbi példa igazol (M elég nagy).

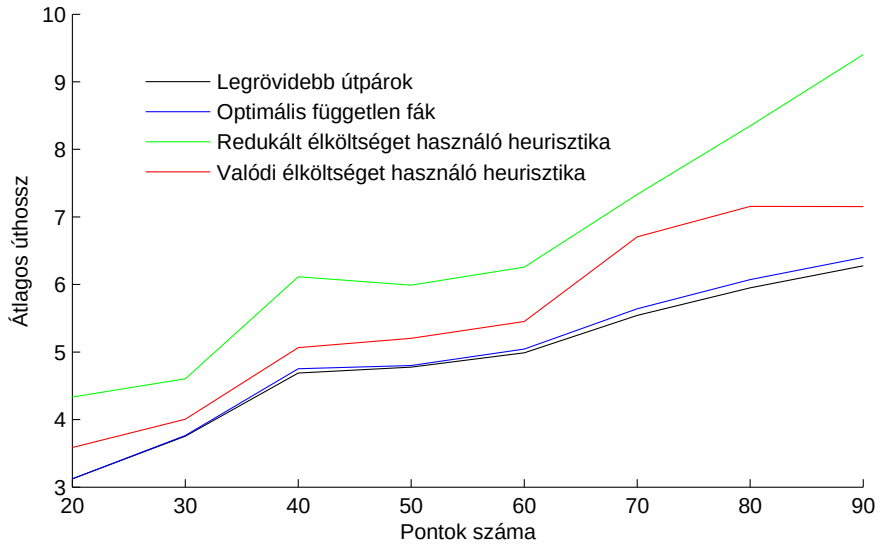


3.1. ábra. Példa az 5/3-os hányadosra

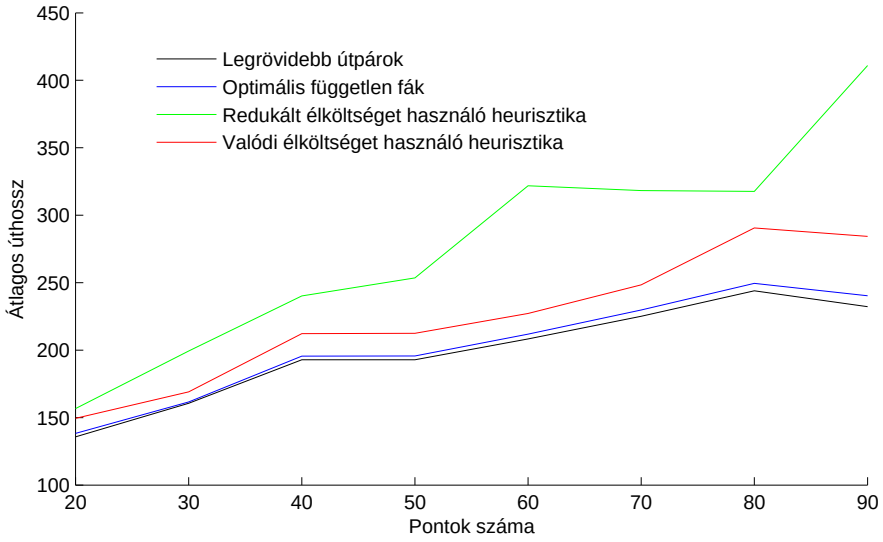
A 3.1 ábrában szokásosan az 1-es csúcs a gyökér, és amelyik élen M van, azt az élet felosztjuk M darab ponttal. Ahol az ábrán nem ér végig két pont között az él, az azt jelenti, hogy a felosztott él utolsó éle már nincs benn a fában. Könnyen végigszámolható, hogy nagyságrendileg a legrövidebb útpárok összhossza $6M^2$, míg a redundáns fákon vett utak összegére minimális érték $10M^2$, tehát a kettő hányadosa 5/3-hoz tart monoton növőleg.

3.3.2. A különböző módszerek összehasonlítása

Megvizsgáltuk, hogy a különböző algoritmusok milyen eredményeket adnak átlagosan véletlen gráfokon, először élköltség nélkül, majd véletlen 1 és 100 közötti élköltséggel. Ezek eredményeit a következő két grafikonban ismertetjük.

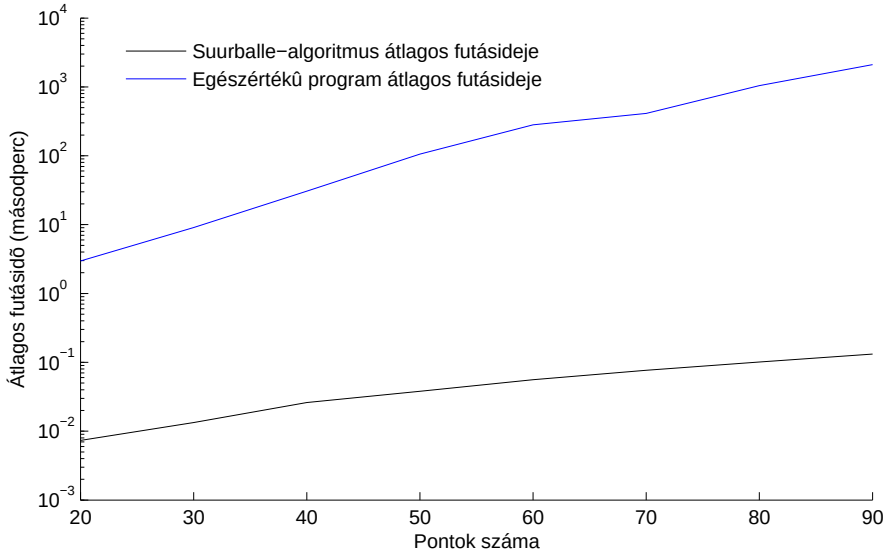


3.2. ábra. Átlagos úthosszak a különböző módszerekkel véletlen egységnyi élköltségű gráfokon



3.3. ábra. Átlagos úthosszak a különböző módszerekkel véletlen gráfokon véletlen, 1 és 100 közötti élköltséggel

Látszik a grafikonokon is, hogy az optimális független fapár értéke szinte megegyezik a legrövidebb útpárok értékével. Továbbá megmutatjuk, hogy az egészértékű program futásideje mennyire lassú a Suurballe-algoritmus futásidejéhez képest.



3.4. ábra. Átlagos futásideje a Suurballe-algoritmusnak és az egészértékű programnak véletlen gráfokon

Irodalomjegyzék

[1] Suurballe, J. W., *Disjoint paths in a network*, Networks 4 (1974), 125-145

[2] Suurballe, J. W., Tarjan, R. E., *A quick method for finding shortest pairs of disjoint paths*, Networks 14 (1984), 325-336

[3] Muriel Médard, Steven G. Finn, Richard A. Barry, Robert G. Gallager, *Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs*, (1999)

[4] Weiyi Zhang, Guoliang Xue, Jian Tang, Krishnaiyan Thulasiraman, *Faster Algorithms for Constructing Recovery Trees Enhancing QoP and QoS*, (2008)

[5] A. Itai, R. Rodeh, *The multi-tree approach to reliability in distributed networks*, Proc. 25th Annu. IEEE Symp. on Foundations of Computer Sciences (1984), 137-147