Többtermékes folyam problémák

Szakdolgozat

Írta: **Fige Péter**

Témavezetők: **Frank András** egyetemi tanár
Király Zoltán adjunktus

ELTE Természettudományi Kar
Operációkutatási Tanszék

2000

Tartalomjegyzék

1. Bevezetés

A folyamok a való életben felmerülő szállítási problémák széles körének modellezésére alkalmasak. A klasszikus kérdésre, hogy mennyi egy hálózat áteresztőképessége, 1962-ban Ford és Fulkerson adta meg a választ, ezzel elindítva a folyamok elméletét, ami az eredeti egyszerű problémából mára szerteágazó területté vált.

A felmerülő problémák többsége az alkalmazások – nagyrészt a távközlés – felől érkezik. A kérdésekre kapható válaszok egyrészt NP-teljességi, illetve közelíthetőségi tételek, másrészt algoritmusok. A feladat nehézségétől függően ez jelentheti polinomiális algoritmusok keresését, lineáris programozást használó hagyományos módszereket kiegészítő kombinatorikus módszerek kidolgozását, illetve NP-teljes problémákra hatékony közelítő algoritmusok adását.

Jelen dolgozat célja, hogy a távközlésben felmerülő speciális folyam problémák, valamint a megoldásukra született algoritmikus módszerek összefoglalását adja.

A dolgozat felépítése a következő. A 2. fejezetben definiáljuk a többtermékes folyam problémát, röviden összefoglaljuk a korábban ismert eredményeket, definiáljuk az osztatlan folyam problémát. A 3. fejezetben a tört többtermékes folyam probléma megoldására adunk közelítő algoritmust általános gráfban, illetve egzakt, gyors algoritmus körgyűrű esetén. Az osztatlan folyam problémát a 4. fejezetben tárgyaljuk. Közelítő algoritmust adunk speciális esetekben, és egy egyszerű alkalmazást ismertetünk.

Végezetül a szerző szeretne köszönetet mondani témavezetőinek, Frank Andrásnak és Király Zoltánnak segítségükért és értékes tanácsaikért. A szakdolgozat témájához kapcsolódó kutatásokat az Egyetemközi Távközlési és Informatikai Központ (ETIK) támogatta.[1]

[1]Az ETIK ipari partnerei: Ericsson Magyarország Kft, KFKI Számítástechnikai Rt, Magyar Távközlési Rt, Sun Microsystems Magyarország Kft, Westel 900 GSM Mobil Távközlési Rt.

2. Alapfogalmak

2.1. Az egy és többtermékes folyam probléma

2.1. Definíció. *Legyen $G = (V, E)$ irányított gráf, $s, t \in V$ kijelölt pontjai, a forrás és a nyelőpont. Ekkor egy $f : V \to R^+$ függvény $s \to t$-folyam, ha $\forall v \in V - \{s, t\}$-re $\sum_{(uv)=e\in E} f(e) = \sum_{(vu)=e\in E} f(e)$, azaz minden s-től és t-től különböző pontban a bemenő éleken a folyamösszeg megegyezik a kimenő éleken a folyamösszeggel. Az f folyam nagysága $\sum_{(sv)=e\in E} f(e)$.*

2.2. Definíció. *Az f folyam megengedett a $c : E \to R^+$ kapacitásfüggvényre nézve, ha $\forall e \in E$-re $f(e) \leq c(e)$, azaz minden élen a folyam értéke legfeljebb az élkapacitás.*

A legkorábban vizsgált feladat az alábbi:

Maximális folyam probléma
Adott: $G = (V, E)$ gráf, $s, t \in V$ pontok, és $c : E \to R^+$ kapacitásfüggvény.
Feladat: Adjunk meg egy maximális nagyságú megengedett $s \to t$-folyamot!

Ebben a feladatban egy *terméket* kellett eljuttatnunk az s pontból a t pontba, így a továbbiakban az egytermékes folyam probléma elnevezést is használjuk. A feladat kézenfekvő általánosítása, ha ugyanabban a gráfban több terméket szeretnénk egyszerre szállítani:

Többtermékes folyam probléma
Adott: $G = (V, E)$ gráf, élein c kapacitásfüggvény, valamint k darab termék, mindegyikhez egy $s_i, t_i \in V$ forrás és nyelő, az i. termék nagysága $d_i \in R^+$.
Feladat: Adjunk meg minden termékre egy d_i nagyságú f_i $s_i \to t_i$-folyamot, úgy, hogy ezek összege ne sértse a kapacitásfüggvényt: $\forall e \in E$-re $f(e) := = \sum_{i=1}^{k} f_i(e) \leq c(e)$.

Egy – terméket jelölő – s, t, d hármasra a továbbiakban az *igény* elnevezést is használjuk, illetve többtermékes folyam helyett ekvivalens jelentéssel használjuk a *törtfolyam* szót.

2.2. A vágásfeltétel

A maximális folyam problémára először Ford és Fulkerson adott megoldást, tőlük származik a következő alapvető tétel:

2.3. Definíció. *Egy $S \subset V$ halmaz $s \to t$-vágás, ha $s \in S$ és $t \notin S$. A vágás értéke $\sum(c(e) : (uv) = e \in E, u \in S, v \notin S)$.*

2.4. Tétel. *A maximális folyam nagysága megegyezik a minimális $s \to t$-vágás nagyságával.* □

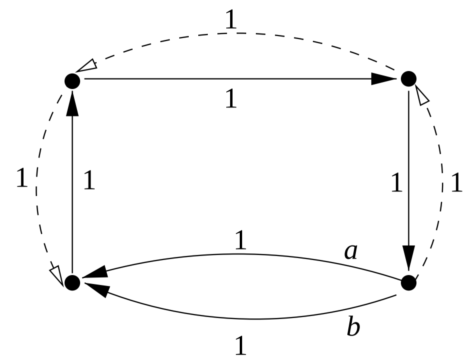
A tétel úgy is megfogalmazható, hogy pontosan akkor létezik d nagyságú $s \to t$-folyam, ha minden $s \to t$-vágás legalább d nagyságú. Ez a megfogalmazás általánosítható többtermékes folyamokra is:

2.5. Definíció. *Egy többtermékes folyam probléma teljesíti a vágásfeltételt, ha minden $S \subset V$ vágásra a vágás értéke legalább $\sum(d_i : s_i \in S, t_i \notin S)$*

A vágásfeltétel szükséges a többtermékes folyam létezéséhez, de nem elégséges, amint az 1. ábra mutatja. Az ábrán – és a továbbiakban – a tömör élek jelölik a gráf éleit, a szaggatott élek pedig az úgynevezett *igénygráfot*. Amennyiben az u pontból a v pontba d mennyiséget kell szállítani, akkor az igénygráfba behúzunk egy d nagyságú $u \to v$ *igényélet*. Az gráf élei mellé írt számok az él kapacitását jelentik, míg az igényélek mellé írt számok az igény nagyságát jelölik.

Amint látható az ellenpéldában minden él kapacitása 1, és mindhárom igény is 1 nagyságú. Egyszerűen ellenőrizhető, hogy a vágásfeltétel ebben az esetben teljesül, azaz – az igénygráf segítségével megfogalmazva – ha bárhogy kijelölünk néhány pontot, a kijelölt halmazból kilépő gráfélek kapacitásának összege legalább annyi, mint a halmazból kilépő igényélek nagyságának összege. Például a bal felső és a jobb alsó pontot kijelölve, ebből a halmazból három gráf él lép ki (a vízszintesek), és csak két igényél (a két függőleges).

Az is könnyen látható, hogy mindhárom terméknek használnia kell az a, vagy b élek valamelyikét, hiszen a gráfban csak az óramutató járásával megegyező irányban lehet „haladni". Emiatt nem létezhet megengedett törtfolyam, mert az a és b élen összesen 3 egységnyi folyam kell, hogy áthaladjon, míg az összkapacitásuk csak 2.



1. ábra: A vágásfeltétel nem elég a többtermékes folyam létezéséhez

2.3. Lineáris programozási megfogalmazás

Mielőtt szükséges és elégséges feltételt adnánk a törtfolyam létezésére, írjuk fel a többtermékes folyam problémát, mint lineáris programozási feladatot.

Az alábbi megfogalmazásban az $f_i(e)$ változó adja meg az i. termékhez tartozó folyam értékét az e élen.

$$\begin{array}{lr}
i = 1, ..., k & \\
\forall v \in V - \{s_i, t_i\} & \displaystyle\sum_{e=(uv)\in E} f_i(e) - \sum_{e=(vu)\in E} f_i(e) = 0 \quad (1) \\
i = 1, ..., k & \displaystyle\sum_{e=(s_i u)\in E} f_i(e) - \sum_{e=(u s_i)\in E} f_i(e) = d_i \quad (2) \\
\forall e \in E & \displaystyle\sum_{i=1}^{k} f_i(e) \le c(e) \quad (3) \\
i = 1, ..., k & \\
\forall e \in E & f_i(e) \ge 0 \quad (4)
\end{array}$$

Ezek a feltételek a többtermékes folyam definíciójának közvetlen átírásából származnak. (1) biztosítja, hogy az f_i függvények folyamot határozzanak meg, (2) miatt a folyam nagysága pont d_i lesz, és (3) biztosítja, hogy a folyamok összege megengedett, azaz nem sérti a kapacitásokat. A $f_i(e)$ változók nemnegativitását (4) garantálja.

Felhasználva a következő közismert állítást, egy másik leírását is adhatjuk a problémának, ami bizonyos esetekben hasznosabb az előzőnél.

2.6. Állítás. *Egy $s \to t$-folyam felbontható legfeljebb élszámnyi $s \to t$-út összegére. ($s \to t$-út alatt itt olyan speciális $s \to t$-folyamot értünk, amelyben egyetlen s-ből t-be vezető út mentén vannak nem 0 értékű élek. Ebből az is következik, hogy ezen nem 0 élek értéke ugyanaz.)* □

Jelölje tehát $\mathcal{P}_{uv}$ az u-ból v-be vezető utak halmazát és legyen $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_{s_i t_i}$. Rendeljünk minden $P \in \mathcal{P}$ úthoz egy $x(P)$ változót. Ekkor a többtermékes folyam probléma a következőképpen fogalmazható meg.

$$\begin{array}{lr}
i = 1, ..., k & \displaystyle\sum_{P\in\mathcal{P}_{s_i t_i}} x(P) = d_i \quad (5) \\
\forall e \in E & \displaystyle\sum_{e\in P\in\mathcal{P}} x(P) \le c(e) \quad (6) \\
\forall P \in \mathcal{P} & x(P) \ge 0 \quad (7)
\end{array}$$

A leírásban (7) miatt a változók nemnegatívak lesznek, (5) szerint az egyes termékekhez tartozó folyamok mérete pontosan az igény nagysága, míg (6) miatt az eredményül kapott folyam nem sérti az élkapacitásokat.

Szembetűnő ennek a megfogalmazásnak a rendkívül egyszerű alakja. Ez annak köszönhető, hogy utak összege automatikusan folyam lesz, azaz teljesíti az (1)-ben megfogalmazott megmaradási feltételt. Az egyszerűséggel szemben azonban hátrány, hogy az utak – és így a változók – száma a gráf pontszámában exponenciális is lehet, ami miatt hagyományos lineáris programozási módszerekkel a feladat eme formája kezelhetetlen. Mégis használják ezt a formát a gyakorlatban, de akkor az úgynevezett oszlopgenerálásos módszerrel együtt.

2.4. A távolságfeltétel

A 2.2. fejezetben láttuk, hogy a vágásfeltétel nem elégséges a törtfolyam létezéséhez. A példában egy egyszerű gondolattal sikerült megmutatnunk, hogy nem létezik többtermékes folyam. Kijelöltünk néhány élet (az alsó kettőt), megmutattuk, hogy minden terméknek használnia kell ezeket az éleket, végezetül láttuk, hogy a kijelölt élek kapacitása kisebb mint a rajtuk áthaladni kényszerülő termékek össznagysága. Ez a gondolat a következő általános feltételnek egy speciális esete:

2.7. Definíció. *Egy $G = (V, E)$ gráf c kapacitásfüggvénnyel, és s_i, t_i, d_i $(i = 1, ..., k)$ igényekkel kielégíti a távolságfeltételt, ha minden $w : E \to R^+$ költségfüggvényre*

$$\sum_{e\in E} w(e)c(e) \geq \sum_{i=1}^{k} l_w(s_i, t_i)d_i$$

ahol $l_w(s_i, t_i)$ a w költségfüggvény szerinti legrövidebb $s_i \to t_i$ út.

Az egyszerűbb írásmód miatt vezessük be a $w(P)$ jelölést, ami a P út költségét jelenti, azaz $w(P) = \sum_{e\in P} w(e)$.

2.8. Állítás. *Amennyiben létezik törtfolyam, akkor a távolságfeltétel teljesül.*

Bizonyítás. Legyen tehát f_i az i. termékhez tartozó $s_i \to t_i$-folyam, úgy, hogy ezek összege megengedett. Bontsunk fel minden f_i-t utak összegére, ebből kapunk $x(P)$ értékeket amelyek az (5), (6), (7) egyenleteket kielégítik.

Legyen $w : E \to R$ tetszőleges súlyfüggvény, és jelölje P'_{uv} a w szerint vett legrövidebb utat u-ból v-be. (A 2.7. definíció jelölése szerint ekkor $w(P'_{uv}) = l_w(u, v)$.) Ekkor a folyam megengedettsége miatt $c(e) \geq \sum_{e\in P\in\mathcal{P}} x(P)$, tehát

$$\sum_{e\in E} w(e)c(e) \geq \sum_{e\in E} w(e) \sum_{e\in P\in\mathcal{P}} x(P) = \sum_{P\in\mathcal{P}} x(P) \sum_{e\in P} w(e) =$$
$$\sum_{i=1}^{k} \sum_{P\in\mathcal{P}_{s_i t_i}} x(P)w(P) \geq \sum_{i=1}^{k} w(P'_{s_i t_i}) \sum_{P\in\mathcal{P}} x(P) = \sum_{i=1}^{k} l_w(s_i, t_i)d_i$$

ami pontosan azt jelenti, hogy a távolságfeltétel teljesül. □

Nem véletlen, hogy a bizonyítás során a lineáris programozási felírás változóit és gondolatait használtuk fel, ugyanis a távolságfeltétel a törtfolyam létezésének duálisa:

2.9. Tétel. *Pontosan akkor létezik megengedett törtfolyam, ha teljesül a távolságfeltétel.*

Bizonyítás. Az (5)-beli egyenletekhez vezessük be a $y(i)$ duális változókat ($i = 1, ..., k$), a (6) megengedettségi feltételhez pedig a $w(e)$ ($e \in E$) változókat. Ekkor a Farkas lemma szerinti duális:

$$i = 1, ..., k \quad \forall P \in \mathcal{P}_{s_i t_i} \qquad y(i) + \sum_{e \in P} w(e) \geq 0 \tag{8}$$

$$\sum_{i=1}^{k} y(i)d(i) + \sum_{e \in E} w(e)c(e) < 0 \tag{9}$$

$$\forall e \in E \qquad w(e) \geq 0 \tag{10}$$

Tegyük fel, hogy nem létezik megengedett törtfolyam. Ekkor létezik y és w, ami a fenti duálisnak megoldása. (8) szerint minden $P \in \mathcal{P}_{s_i t_i}$-re $-$ $-y(i) \leq w(P)$, azaz $-y(i) \leq l_w(s_i, t_i)$. Legyen $y'(i) := -l_w(s_i, t_i)$. Ez a duálnak szintén megoldása (w változatlan), hiszen $y'(i) \leq y(i)$ ami miatt (8) baloldala nem csökken, míg (9) baloldala nem nő.

Ekkor viszont (9) nem más mint

$$\sum_{e \in E} w(e)c(e) < \sum_{i=1}^{k} -y(i)d_i = \sum_{i=1}^{k} l_w(s_i, t_i)d_i$$

vagyis találtunk w-t ami megsérti a távolságfeltételt.

A másik irányt – miszerint a törtfolyam létezéséből következik a távolságfeltétel teljesülése – a 2.8. állításban már igazoltuk. $\square$

2.5. Irányítatlan gráfok

Irányítatlan gráfokra a fenti definíciók automatikusan átmennek. A keletkező folyamot irányított értelemben tekintjük, de az élkapacitásba az élen mindkét irányban áthaladó termékek beleszámítanak.

Az alábbiakban felsoroljuk az eddigi definíciók irányítatlan változatát. Legyen $G = (V, E)$ irányítatlan gráf, és adjunk az éleinek egy tetszőleges referenciairányítást, azaz minden élet irányítsunk meg tetszőlegesen. Jelölje ezt a gráfot is G.

2.10. Definíció. *Legyen $G = (V, E)$ irányítatlan gráf, $s, t \in V$ kijelölt pontjai, a forrás és a nyelőpont. Ekkor egy $f : V \to R$ függvény irányítatlan $s \to t$-folyam, ha $\forall v \in V - \{s, t\}$-re $\sum_{(uv)=e \in E} f(e) = \sum_{(vu)=e \in E} f(e)$, azaz minden s-től és t-től különböző pontban a bemenő éleken a folyamösszeg megegyezik a kimenő éleken a folyamösszeggel. Az f folyam nagysága $\sum_{(sv)=e \in E} f(e)$.*

2.11. Definíció. *Az f folyam megengedett a $c : E \to R^+$ kapacitásfüggvényre nézve, ha $\forall e \in E$ $|f(e)| \leq c(e)$.*

A definíció tehát csak annyiban különbözik az irányított esettől, hogy a folyam értéke negatív is lehet, azaz nem kötjük meg, hogy az éleken csak „előre" mehet folyam: $f(uv) = -10$ azt jelenti, hogy az uv élen v-ből u-ba 10 egységnyi a folyam. Emiatt azonban a megengedettség feltételében abszolút értékben kell kezelni a folyamot. Vegyük észre, hogy bár a terminológiában megtartottuk a *forrás* és *nyelő* kifejezéseket, a definíció érzéketlen ezek felcserélésére. Egy megengedett $s \to t$-folyam -1-szerese ugyanakkora nagyságú megengedett $t \to s$-folyam.

Irányítatlan többtermékes folyam probléma
Adott: $G = (V, E)$ irányítatlan gráf, élein c kapacitásfüggvény, valamint k darab termék, mindegyikhez egy $s_i, t_i \in V$ forrás és nyelő, az i. termék nagysága $d_i \in R^+$.
Feladat: Adjunk meg minden termékre egy d_i nagyságú f_i irányítatlan $s_i \to t_i$-folyamot, úgy, hogy ezek összege ne sértse a kapacitásfüggvényt: $\forall e \in E$-re $f(e) := \sum_{i=1}^{k} |f_i(e)| \leq c(e)$.

2.6. Egész értékű folyamok

2.12. Definíció. *Egy f folyam egész, ha minden $e \in E$ élre $f(e)$ egész. Egy többtermékes folyam egész, ha minden termékre a hozzá tartozó folyam egész.*

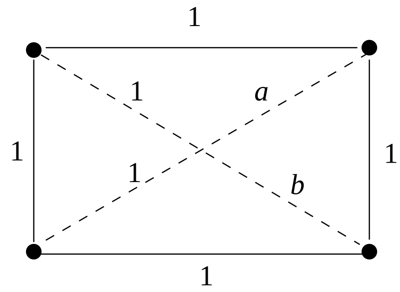
Az egytermékes folyam problémát megoldó algoritmus érdekes tulajdonsága, hogy amennyiben az élkapacitások egészek, akkor automatikusan egész folyamot talál.

Több termék esetén ez nem garantálható, amint az a 2. ábrán látható. Ez a gráf irányítatlan, minden élkapacitás 1, és mindkét igény 1 nagyságú. Tört folyam létezik, mégpedig az igények egyik felét az egyik lehetséges úton visszük, a másik felét pedig a másik úton. (Azaz minden élre $f_a(e) = f_b(e) = \frac{1}{2}$.)

Mivel az igények nagysága 1, ezért a két termékhez rendelt egész folyam ez esetben csak egy-egy út lehet. A szimmetria miatt feltehető, hogy az a termék a felső és bal oldali élet használja. Ekkor azonban a b terméknek nem találhatunk utat, mivel a bal felső pontból kiindulo éleken már nincs szabad kapacitás, azaz egész folyam nem létezhet.

Felvetődik a kérdés, hogy mikor létezik egész többtermékes folyam. Ez a kérdés nem könnyű, egész pontosan:

2.13. Tétel. *Annak megállapítása, hogy létezik-e egész többtermékes folyam NP-teljes.* □



2. ábra: Több termék esetén nincs mindig egész megoldás

2.7. Osztatlan folyamok

Az előző fejezet ellenpéldájában (2. ábra) azt használtuk ki, hogy egy 1 nagyságú egész folyam valójában egy út. Sok alkalmazásban fontos megszorítás, hogy a termékeket egyetlen út mentén kell szállítani. Ennek modellje az osztatlan folyam probléma:

Osztatlan folyam probléma
Adott: $G = (V, E)$ irányított gráf c kapacitásfüggvénnyel, s_i, t_i, d_i igények.
Feladat: Találjunk többtermékes folyamot, amelyben az i. termékhez rendelt f_i folyam egy $s_i \to t_i$-út.

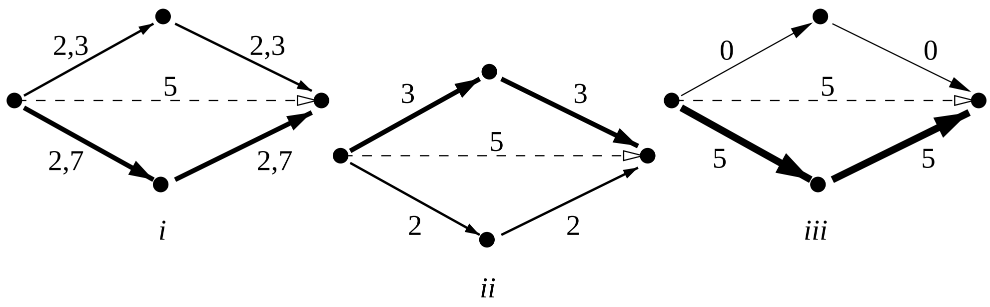
Úgy is fogalmazhatunk, hogy minden i-re az i. igényhez ki kell jelölni egy utat s_i-től t_i-be, és ezen út mentén lefoglalni d_i élkapacitást. Mindezt úgy kell megtennünk, hogy a termékek számára lefoglalt kapacitások ne lépjék túl az élkapacitást.

Amennyiben az igénynagyságok (d_i-k) egészek, akkor egy osztatlan folyam automatikusan egész folyam is lesz. Az osztatlan folyam definíciója azonban akkor is értelmes, ha az igénynagyságok és élkapacitások tetszőleges valós számok.

A három folyamtípus (tört, egész, osztatlan) közti különbséget a 3. ábrán láthatjuk. A folyam ez esetben egyetlen – 5 nagyságú – terméket szállít. Az élkapacitások nincsenek feltüntetve, a vastag élek jelentik a folyam által használt éleket, az élek mellé írt számok pedig a folyam nagyságát.

2.8. Közelítő algoritmusok

Tegyük fel, hogy egy olyan feladatunk van, amelynek minden M megoldásához hozzárendelhető egy $f(M)$ szám, amely a megoldás minőségét méri. Lehet ez egy folyam esetén a teljesített igények mennyisége, a megoldás költsége, vagy bármely más mérőszám. Ezekben az esetekben olyan algoritmust szeretnénk, amely az optimális megoldást adja. Ez a feladattól függően vagy azt jelenti, hogy $f(M)$-et minimalizálni akarjuk (például $f(M)$ az M költsége), vagy azt, hogy maximalizálni ($f(M)$ az M által teljesített igények



3. ábra: i: tört folyam, ii: egész folyam, iii: osztatlan folyam

száma.) Rögzítsünk most egy konkrét feladatot, és tegyük fel a továbbiakban, hogy $f(M)$-et minimalizálni szeretnénk.

Jelölje $A(I)$ az A algoritmus által adott megoldást az I bemeneti adat (input) esetén. Jelölje $OPT(I)$ az optimális megoldását a feladatnak az I input esetén. Ezekkel a jelölésekkel az A algoritmus akkor találja meg az optimumot, ha minden I inputra $f(A(I)) = f(OPT(I))$.

2.14. Definíció. *Azt mondjuk, hogy A c-közelítő algoritmus, ha minden I inputra $f(A(I)) \leq cf(OPT(I))$, azaz az algoritmus által adott megoldás legfeljebb c-szerese az optimumnak.*

Sok NP-teljes probléma van, amelyre létezik c-közelítő algoritmus valamely c-re. Van olyan NP-teljes probléma is, amelyről bebizonyítható, hogy nem lehet jól approximálni, például nem létezhet rá 2-közelítő algoritmus.

Érdekesek még azok a problémák, amelyekre bármely $\epsilon > 0$-ra létezik $(1+$ $+\,\epsilon)$-közelítő algoritmus. Természetesen a jobb közelítésnek ára van, minél kisebb ϵ-t választunk, annál nagyobb lesz a futási idő.

2.15. Definíció. *Azt mondjuk, hogy egy problémára létezik ϵ-közelítő algoritmus séma, ha minden rögzített $\epsilon > 0$-ra létezik $(1 + \epsilon)$-közelítő algoritmus, amelynek futási ideje polinomiális n-ben, ahol n a probléma mérete.*

2.16. Definíció. *Azt mondjuk, hogy egy problémára létezik teljesen polinomiális ϵ-közelítő algoritmus séma, ha minden $\epsilon > 0$-ra létezik $(1 + \epsilon)$-közelítő algoritmus, amelynek futási ideje polinomiális $1/\epsilon$-ban és n-ben, ahol n a probléma mérete.*

A közelítő algoritmusok nem csak akkor lehetnek érdekesek, ha a feladat NP-teljes. Vannak problémák, amelyekre van egzakt polinomiális algoritmus, de a gyakorlatban felmerülő problémaméretek esetén ez lassúnak bizonyul. Ekkor lehet fontos egy gyors közelítő algoritmus, amely majdnem optimális megoldást ad.

A közelítő algoritmusok és algoritmus sémák részletesebb leírása sok híres NP-teljes feladattal együtt megtalálható [4]-ben.

3. Tört folyamok

A 2.2 fejezetben láttunk szükséges és elégséges feltételt a többtermékes folyam létezésére. Mivel a problémát megfogalmaztuk LP-ként, ezért automatikusan egy polinomiális algoritmust is kapunk, hiszen a lineáris programozási feladat polinomiális időben megoldható.

Ennek a megközelítésnek az a hátránya, hogy a keletkező lineáris programozási feladat túl nagy méretű. Például n pont, m él és k igény esetén az (1), (2), (3), (4) szerinti felírásban a változók száma mk a feltételek száma pedig $k(n-2+m)+k+m$. Az (5), (6), (7) szerinti utas megfogalmazásban a változók száma exponenciálisan sok lehet n-ben! A gyakorlatban előforduló nagyobb feladatméretek esetén ez azt jelenti, hogy a program futási ideje túl nagy, lineáris programozási módszerekkel nagy feladatok nem kezelhetők.

Ebben a fejezetben két módszert ismertetünk, amellyel nagy feladatok is megoldhatók. Ezek egyike azonban csak közelítő algoritmust ad, a másik pedig egy speciális gráfosztályon működik csak, azonban minimális költségű folyamot ad.

3.1. Közelítő algoritmusok a többtermékes folyam problémára

A gyakorlatban előforduló problémákat megoldó algoritmusok gyakran használnak többtermékes folyam problémát megoldó szubrutint, amelyet rengetegszer végre kell hajtani az algoritmus futása során. Ilyen esetekben fontos, hogy a többtermékes folyam megkeresése nagyon gyorsan menjen. Mivel az alapprobléma legtöbbször NP-teljes és közelítő megoldást akarunk találni, a megoldás minőségét nem rontja, ha a többtermékes folyam problémát csak közelítőleg oldjuk meg, cserébe viszont a futás nagyon gyors lesz.

A következő részben a folyam probléma két változatának egy olyan algoritmikus megközelítését ismertetjük, amely egy teljesen polinomiális ϵ-közelítő algoritmus sémát ad, és futási időben jobb a lineáris programozási módszereknél.

3.1.1. Az algoritmusok alapgondolata

A maximális folyam probléma esetén Ford és Fulkerson egytermékes folyamalgoritmusa úgy működik, hogy a gráfban lévő szabad kapacitásokon próbál még több folyamot elvezetni, így a közbenső lépésekben lévő folyam (primál megoldás) mindig megengedett, nem sérti az élkapacitásokat. A növelések során az algoritmus eljut egy olyan pontra, ahol már nincs lehetőség további növelésre. Ekkor megkonstruál egy – a folyamnagysággal megegyező értékű – duál megoldást, ami bizonyítja az optimalitást, azaz a folyam maximalitását.

A most következő – Garg és Könemann [1] által adott – algoritmusok sajátossága, hogy a futás során a primál megoldás nem megengedett, azaz a folyam megsértheti, sőt jócskán meg is sérti a kapacitás-feltételeket. A duál megoldás sem megengedett kezdetben, de egyre kevésbé sérti a feltételeket. Mikor megengedetté válik, akkor a folyamot leosztjuk úgy, hogy már ne sérüljenek a kapacitáskorlátok, és az így kapott folyamról megmutatható, hogy értéke közel van az optimálishoz.

3.2. A maximális többtermékes folyam probléma

A fent vázolt újszerű megközelítést először nem a többtermékes folyam problémára mutatjuk be, hanem egy másik problémára, amely a maximális folyam probléma többtermékes általánosítása.

Maximális többtermékes folyam probléma
Adott: $G = (V, E)$ irányított gráf, c kapacitásfüggvény, s_i, t_i $i = 1, ..., k$ pontpárok.
Feladat: Adjuk meg az f_i $s_i \to t_i$-folyamokat úgy, hogy a folyamok összege megengedett legyen c-re nézve, és a folyamok nagyságának összege maximális legyen.

A probléma lineáris programozási megfogalmazása lényegében (5), (6), (7)-el egyezik meg, csak a folyamértékekre vonatkozó megkötést kell elhagynunk, és a folyamösszeget, mint célfüggvényt kell beépítenünk. Jelölje most is $\mathcal{P}_{s_i t_i}$ az s_i-ből t_i-be menő utakat, és legyen $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_{s_i t_i}$.

$$\forall e \in E \qquad \sum_{e \in P \in \mathcal{P}} x(P) \leq c(e)$$
$$\forall P \in \mathcal{P} \qquad x(P) \geq 0$$

$$\max \sum_{P \in \mathcal{P}} x(P)$$

A probléma duálisában az élekre egy l nemnegatív hosszfüggvényt írunk. Jelölje a P út l szerinti hosszát $l(P)$. Ekkor a duális:

$$\forall P \in \mathcal{P} \qquad l(P) \geq 1$$
$$\forall e \in E \qquad l(e) \geq 0$$

$$\min \sum_{e \in E} l(e)c(e)$$

Legyen $r_i(l) := \min_{P \in \mathcal{P}_{s_i t_i}} l(P)$ az s_i-ből t_i-be menő legrövidebb út hossza, és legyen $\alpha(l) := \min_{i=1,...,k} r_i(l)$ a legrövidebb út, ami egy igénypárt köt össze. Legyen továbbá $D(l) := \sum_{e \in E} l(e)c(e)$. A duális tehát ezen jelölésekkel így hangzik: az $\alpha(l) \geq 1$ feltétel mellett minimalizáljuk $D(l)$-t. Úgy is fogalmazhatunk, hogy a $\beta := \min_l D(l)/\alpha(l)$ minimumot akarjuk meghatározni.

3.2.1. Az algoritmus

Az algoritmus iterációkat végez, minden iterációban növeli a folyamot, majd növeli az l duális megoldást. Az alábbi leírásban szereplő δ és ϵ értékeket később definiáljuk.

`MMCF algoritmus`

- `Legyen minden` e `élre` $l(e) := \delta$.
- `Legyen minden` $P \in \mathcal{P}$ `útra` $x(P) := 0$.
- `Amíg` $\alpha(l) < 1$ `ismételjük a következő lépéseket:`
  - `Számoljuk ki` $\alpha(l)$`-et, legyen` P `legrövidebb út, azaz` $l(P) = \alpha(l)$.
  - `Legyen` u `a minimális élkapacitás` P `élein az eredeti kapacitásfüggvény szerint:` $u := \min_{e \in P} c(e)$.
  - `Növeljük` $x(P)$`-t` u`-val.`
  - `A` P `út minden` $e \in P$ `élére legyen` $l(e) := l(e)(1 + \epsilon \frac{u}{c(e)})$.

3.2.2. Az algoritmus elemzése

Feltehetjük, hogy a c kapacitásfüggvény értéke minden élen legalább 1, ellenkező esetben ugyanis c-t alkalmas számmal felszorozva ez elérhető (a kapott megoldást persze vissza kell osztani az algoritmus futtatása után.) A feltevés mellett, ha $\alpha(l) \geq 1$, akkor $D(l) \geq 1$ is fennáll.

Jelölje l_{i-1} a hosszfüggvényt, f_{i-1} pedig az aktuális folyam által elszállított termékek összegét ($\sum_P x(P)$) az i. iteráció kezdetén. Legyen az egyszerűség kedvéért $\alpha(i) = \alpha(l_i)$ és $D(i) = D(l_i)$. Ekkor minden $i \geq 1$ iterációra:

$$\begin{aligned}D(i) = \sum_e l_i(e)c(e) = \sum_e l_{i-1}(e)c(e) + \epsilon \sum_{e \in P} l_{i-1}(e)u = \\ D(i-1) + \epsilon(f_i - f_{i-1})\alpha(i-1)\end{aligned}$$

Mivel β az optimum, ezért $\beta \leq \frac{D(i-1)}{\alpha(i-1)}$. Ezt behelyettesítve:

$$D(i) \leq D(i-1) + \epsilon(f_i - f_{i-1})\frac{D(i-1)}{\beta} = D(i-1)\left(1 + \frac{\epsilon}{\beta}(f_i - f_{i-1})\right)$$

amiből adódik, hogy

$$D(i) \leq D(0)\prod_{j=1}^{i}\left(1 + \frac{\epsilon}{\beta}(f_j - f_{j-1})\right) \leq (\delta m)e^{\frac{\epsilon}{\beta}\sum_{j=1}^{i}(f_j - f_{j-1})} = (\delta m)e^{\epsilon f_i/\beta}$$

Tudjuk, hogy az algoritmus utolsó – t-edik – fázisa után $\alpha(t) \geq 1$ és így az elemzés elején tett feltevésünk szerint $D(t) \geq 1$. Ebből pedig az utolsó fázisra $1 \leq D(t) \leq (\delta m)e^{\epsilon f_t/\beta}$ következik, amit átrendezve (mint később látni fogjuk $\delta m < 1$):

$$\frac{\beta}{f_t} \leq \frac{\epsilon}{-\ln(\delta m)} \tag{11}$$

3.1. Állítás. *Létezik* $\frac{f_t}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}$ *nagyságú megengedett folyam.*

Bizonyítás. Osszuk le az algoritmus által adott megoldást $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$-val! Megmutatjuk, hogy az így kapott folyam megengedett. Legyen ugyanis e tetszőleges él. Mikor utoljára növeltük meg l-et az e élen, akkor e egy 1-nél kisebb összhosszú úton volt rajta, így $l(e) < 1$ volt. Mivel minden növelés legfeljebb $(1+\epsilon)$-szoros, így $l_t(e) < 1+\epsilon$, azaz az algoritmus futása során az e hossza legfeljebb $\frac{1+\epsilon}{\delta}$-szorosára nő.

Tegyük fel, hogy az i. fázisban az e terhelése $a_i c(e)$-vel nőtt, ahol $0 \leq \leq a_i \leq 1$. Mivel $1 + a_i\epsilon \geq (1+\epsilon)^{a_i}$, ezért

$$\frac{1+\epsilon}{\delta} \geq \prod_i (1 + a_i\epsilon) \geq (1+\epsilon)^{\sum_i a_i}$$

és így az él terhelése az algoritmus végén $c(e)\sum_i a_i \leq c(e)\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$, a leosztott folyam tehát megengedett. □

3.2. Tétel. *Ha* $0 < \epsilon' < 1$, *akkor létezik* $O(m\lceil\frac{1}{\epsilon'}\log_{1+\epsilon'} m\rceil T_{sp})$ *futási idejű* $(1+\epsilon')$*-közelítő algoritmus a maximális többtermékes folyam problémára. Itt* T_{sp} *a legrövidebb út megtalálásának ideje,* m *pedig az élek száma.*

Bizonyítás. A maximális folyamérték β, a fent ismertetett algoritmus által adott folyamérték $f_t/\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. E két érték hányadosa legyen γ. Használva a (11)-es egyenlőtlenséget kapjuk, hogy

$$\gamma \leq \frac{\epsilon \log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{-\ln(\delta m)} = \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln\frac{1+\epsilon}{\delta}}{\ln\frac{1}{\delta m}}$$

Válasszuk a δ értéket $(1+\epsilon)((1+\epsilon)m)^{-1/\epsilon}$-nak. Ekkor $\frac{\ln(1+\epsilon)\delta^{-1}}{-\ln(\delta m)} = \frac{1}{1-\epsilon}$, tehát

$$\gamma \leq \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)} \leq \frac{\epsilon}{(1-\epsilon)(\epsilon - \epsilon^2/2)} \leq \frac{1}{(1-\epsilon)^2}$$

Legyen tehát $\epsilon := \epsilon'/4$, ekkor $\gamma \leq 1 + \epsilon'$.

Minden iterációban a kiválasztott P út minimális kapacitású élének hosszát $(1+\epsilon)$-szorosára növeljük, így egy él kevesebb, mint $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ iterációban lehet a választott út minimális éle. Mivel

$$\log_{1+\epsilon}\frac{1+\epsilon}{\delta} = \log_{1+\epsilon}((1+\epsilon)m)^{1/\epsilon} = 1/\epsilon(1 + \log_{1+\epsilon} m)$$

így legfeljebb $m\lceil\frac{1}{\epsilon}(1+\log_{1+\epsilon} m)\rceil$ iteráció lehet, egy iteráció ideje pedig T_{sp}, ami a gráfban lévő legrövidebb út megtalálásának ideje. (Az, hogy $\epsilon' = 4\epsilon$ csak konstansszorosára növeli a futási időt.) $\square$

Feltéve, hogy a legrövidebb utakat úgy keressük, hogy minden igényre keresünk legrövidebb utat, majd ezek közül a legrövidebbet választjuk, az algoritmus futási ideje $O(\frac{1}{\epsilon^2}km^2\log^2 m)$. Amennyiben a futás során nem a legrövidebb utat keressük, hanem megelégszünk a legrövidebb út egy alkalmas közelítésével, akkor – mint Lisa Fleischer [2]-ben megmutatja – a futási idő $O(\frac{1}{\epsilon^2}m^2\log^2 m)$-re csökkenthető. Ez az eredmény abból a szempontból is érdekes, hogy a futási idő független a termékek számától.

3.3. A többtermékes folyam probléma közelítése

A (5), (6), (7)-al definiált lineáris programozási megfogalmazásnál csak az érdekes, hogy van-e megoldás, vagy nincs. Egy eldöntendő kérdést nem lehet ϵ-közelíteni, ezért egy másik felírást fogunk használni, ami nem követeli meg, hogy az igényeket kielégítsük, mindössze azt, hogy minden igény $0 \le \lambda$-szorosát szállítsa el a folyam. Ennek megfelelően a lineáris program:

$$\begin{array}{ll}
\forall i = 1,...,k & \displaystyle\sum_{P\in\mathcal{P}_{s_it_i}} x(P) \ge \lambda d_i \\
\forall e \in E & \displaystyle\sum_{e\in P\in\mathcal{P}} x(P) \le c(e) \\
\forall P \in \mathcal{P} & x(P) \ge 0
\end{array}$$

$$\max \lambda$$

Ennek duálisa:

$$\begin{array}{ll}
\forall i = 1,...,k, \forall P \in \mathcal{P}_{s_it_i} & \displaystyle\sum_{e\in P} l(e) - y(i) \ge 0 \\
 & \displaystyle\sum_{i=1}^{k} d_i y(i) \ge 1 \\
\forall i = 1,...,k & y(i) \ge 0 \\
\forall e \in E & l(e) \ge 0
\end{array}$$

$$\min \sum_{e\in E} l(e)c(e)$$

Legyen most $r_i(l) := \min_{P\in\mathcal{P}_{s_it_i}} l(P)$, $\alpha(l) := \sum_i r_i(l)d_i$ és $D(l) := = \sum_{e\in E} l(e)c(e)$. Ezzel a jelöléssel a feladat: minimalizáljuk a $D(i)/\alpha(l)$ mennyiséget.

3.3.1. Az algoritmus

Az algoritmus a 3.2.1. részben tárgyalthoz hasonlóan növeli a folyamot, majd növeli az l duális megoldást. A növelés fázisokban történik, egy fázis k darab iterációból áll, az i. iteráció lépésekből áll, egy lépés során az i. termékhez tartozó folyamot növeljük, illetve a súlyfüggvényt módosítjuk. A kezdeti folyam azonosan 0, a kezdeti élhosszok pedig $l(e) = \delta/c(e)$. Az algoritmus a következőképpen néz ki:

`MCF algoritmus`

- `Legyen` $l(e) := \delta/c(e)$.
- `Legyen minden` $P \in \mathcal{P}$ `útra` $x(P) := 0$.
- `Amíg` $D(l) < 1$ `ismételjük:`
  - `Minden` $j = 1, ..., k$ `termékre:`
    - $d' := 0$
    - `Amíg` $D(l) < 1$ `és` $d' < d_j$ `ismételjük:`
      - `Legyen` P `az` l `szerinti legrövidebb út` s_j`-ből` t_j`-be.`
      - `Legyen` $u := \min\{\min_{e \in P} c(e), d_j - d'\}$.
      - `Növeljük` d'`-t és` $x(P)$`-t` u`-val.`
      - `Minden` $e \in P$`-re legyen` $l(e) := l(e)(1 + \frac{\epsilon u}{c(e)})$.

Jelölje $l^{s-1}_{i,j}$ a duális megoldást (hosszfüggvényt) az i. fázis j. iterációjának s. lépésének kezdetén, és legyen $P^s_{i,j}$ a legrövidebb $s_j \to t_j$ út $l^{s-1}_{i,j}$ szerint. Jelölje $d'^{s-1}_{i,j}$ az iteráció első $s-1$ lépésében történt növelések összegét. Az aktuális lépésben megnöveljük a folyamot a $P^s_{i,j}$ út mentén $\min\{d_j - d'^{s-1}_{i,j}, u\}$-val, ahol u a $P^s_{i,j}$ úton lévő legkisebb kapacitás a c kapacitásfüggvény szerint. A hosszfüggvényt megnöveljük $e \in P^s_{i,j}$-re: $l^s_{i,j}(e) := l^{s-1}_{i,j}(e)(1 + \epsilon\frac{u}{c(e)})$, a többi élre $l^s_{i,j}(e) := l^{s-1}_{i,j}(e)$. Végezetül $d'^s_{i,j}$ $d'^{s-1}_{i,j} + u$ lesz. Az iterációnak akkor van vége, mikor $d'_{i,j}$ aktuális értéke d_j lesz.

3.3.2. A közelítés mértéke

Tekintsük az i. fázis j. iterációjának s. lépését, és tegyük fel, hogy ebben az iterációban p darab lépés volt. Ekkor

$$D(l^s_{i,j}) = D(l^{s-1}_{i,j}) + \epsilon \sum_{e \in P^s_{i,j}} l^{s-1}_{i,j}(e) \frac{(d'^s_{i,j} - d'^{s-1}_{i,j})}{c(e)} c(e) =$$

$$= D(l^{s-1}_{i,j}) + \epsilon\, l^{s-1}_{i,j}(P^s_{i,j})(d'^s_{i,j} - d'^{s-1}_{i,j}) \leq D(l^{s-1}_{i,j}) + \epsilon\, l^p_{i,j}(P^p_{i,j})(d'^s_{i,j} - d'^{s-1}_{i,j})$$

ahol az utolsó egyenlőtlenség az l hosszfüggvény monoton növekedése miatt igaz. A p. lépésre a fenti egyenlőtlenség felhasználásával azt kapjuk, hogy

$$D(l_{i,j}^p) \leq D(l_{i,j}^0) + \epsilon\, r_j(l_{i,j}^p) d_j.$$

Legyen $l_{i,j} := l_{i,j}^p$, ahol p az i. fázis j. iterációjának lépésszáma, azaz $l_{i,j}$ az i. fázis j. iterációjának végén adódó hosszfüggvény. Legyen továbbá $D(i) := D(l_{i,k})$ és $\alpha(i) := \alpha(l_{i,k})$. Ezzel a jelöléssel:

$$D(l_{i,k}) \leq D(l_{i,0}) + \epsilon \sum_{j=1}^{k} d_j r_j(l_{i,k})$$

$$D(i) \leq D(i-1) + \epsilon\, \alpha(i). \tag{12}$$

Legyen β a duál optimum, így $\beta \leq \frac{D(i)}{\alpha(i)}$. Ekkor (12)-ből tudjuk, hogy $D(i) \leq \frac{D(i-1)}{1-\epsilon/\beta}$, és felhasználva, hogy $D(0) = \delta m$:

$$D(i) \leq \frac{\delta m}{(1-\epsilon/\beta)^i} = \frac{\delta m}{(1-\epsilon/\beta)}(1 + \frac{\epsilon}{\beta - \epsilon})^{i-1} \leq \frac{\delta m}{1-\epsilon/\beta} e^{\frac{\epsilon(i-1)}{\beta-\epsilon}}$$

Tegyük fel, hogy $\beta \geq 1$. Azt az esetet, amikor ez nem így van, majd külön kezeljük. Ha $\beta \geq 1$, akkor a fenti egyenlőtlenséget tovább becsülve $D(i) \leq$ $\leq \frac{\delta m}{1-\epsilon} e^{\frac{\epsilon(i-1)}{\beta(1-\epsilon)}}$ adódik. Az algoritmus akkor áll le, ha D eléri 1-et. Legyen tehát t az algoritmus utolsó fázisa, ekkor $D(t) \geq 1$, azaz

$$1 \leq D(t) \leq \frac{\delta m}{1-\epsilon} e^{\frac{\epsilon(t-1)}{\beta(1-\epsilon)}}$$

ezt átrendezve (és feltéve, hogy $\frac{1-\epsilon}{\delta m} > 1$) kapjuk, hogy

$$\frac{\beta}{t-1} \leq \frac{\epsilon}{(1-\epsilon) \ln \frac{1-\epsilon}{\delta m}} \tag{13}$$

Az első $t-1$ fázis teljes egészében végrehajtódik, azaz a j. termékből fázisonként d_j-t szállítunk el, így az algoritmus végén a j. termékből legalább $(t-1)d(j)$-t szállít a folyam. Természetesen az így kapott folyam jócskán megsértheti az élkapacitásokat, ezen azonban úgy segítünk, hogy skálázzuk a folyamot:

3.3. Állítás. $\lambda > \frac{t-1}{\log_{1+\epsilon} 1/\delta}$.

Bizonyítás. Az állítás szerint létezik folyam, amely minden termék fenti hányadát elszállítja. Ilyen folyamot úgy kapunk, hogy az algoritmus $t-$ -1-edik lépésében kapott folyamot osztjuk $\log_{1+\epsilon} 1/\delta$-val. Meg kell még mutatnunk, hogy így nem sérülnek az élkapacitások.

Legyen e tetszőleges él. Az e súlya kezdetben $\delta/c(e)$, és mivel $D(t-1) < < 1$, ezért $t-1$ fázis után $l_{t-1,k}(e) < 1/c(e)$, azaz az első $t-1$ fázis során e hossza kevesebb, mint $\frac{1/c(e)}{\delta/c(e)} = \frac{1}{\delta}$-szorosára nőtt.

A 3.1. állításban használt becsléssel adódik, hogy az e élet $c(e)$ nagyságú folyammal megterhelve a hossza legalább $(1+\epsilon)$-szorosára nő, így ha $f(e)$ jelöli az él terhelését $t-1$ fázis után:

$$(1+\epsilon)^{\frac{f(e)}{c(e)}} < \frac{1}{\delta}$$

$$\frac{f(e)}{\log_{1+\epsilon}\frac{1}{\delta}} < c(e)$$

azaz a leosztott folyam már nem sérti a kapacitásfeltételeket, sőt nincs telített él sem, így az állításban szereplő szigorú egyenlőtlenséget is bebizonyítottuk.
□

Ha helyesen választjuk meg az induláskor használt δ számot, akkor az algoritmusunk jó közelítést ad. Jelölje γ az optimális megoldás (β) és az algoritmus által adott megoldás (λ) hányadosát. A 3.3. állítás szerint: $\gamma < < \frac{\beta}{t-1}\log_{1+\epsilon}\frac{1}{\delta}$. Behelyettesítve a (13)-ban kapott becslést:

$$\gamma < \frac{\epsilon \log_{1+\epsilon}\frac{1}{\delta}}{(1-\epsilon)\ln\frac{1-\epsilon}{\delta m}} = \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}\frac{\ln\frac{1}{\delta}}{\ln\frac{1-\epsilon}{\delta m}}.$$

Legyen $\delta = (m/(1-\epsilon))^{-1/\epsilon}$, ekkor $\frac{\ln\frac{1}{\delta}}{\ln\frac{1-\epsilon}{\delta m}} = \frac{1}{1-\epsilon}$, és így

$$\gamma \leq \frac{\epsilon}{(1-\epsilon)^2\ln(1+\epsilon)} \leq \frac{\epsilon}{(1-\epsilon)^2(\epsilon-\epsilon^2/2)} \leq (1-\epsilon)^{-3}.$$

3.3.3. Az algoritmus futási ideje

3.4. Állítás. *Ha $\beta \geq 1$, akkor az algoritmus $\lceil\frac{\beta}{\epsilon}\log_{1+\epsilon}\frac{m}{1-\epsilon}\rceil$ fázis után befejeződik.*

Bizonyítás. Mint láttuk

$$1 \leq \gamma < \frac{\beta}{t-1}\log_{1+\epsilon}\frac{1}{\delta}$$

amiből a fázisok számára azt kapjuk, hogy $t < 1+\beta\log_{1+\epsilon}\frac{1}{\delta}$. Behelyettesítve a választott δ-t az állítás adódik. □

A fázisok száma tehát függ a β-tól. A β értékét tudjuk változtatni az igények skálázásával, amivel elérhetjük, hogy $\beta \geq 1$ teljesüljön, és így az algoritmus elemzésében használt becslések működjenek. Ehhez legyen z_i a maximális megengedett $s_i \to t_i$ folyam értéke a c kapacitásfüggvényre nézve, és legyen $z := \min_i \frac{z_i}{d(i)}$. Világos, hogy $z/k \leq \beta \leq z$.

Osszunk le tehát minden igényt úgy, hogy $z/k = 1$ legyen, az így módosított feladatra $\beta \geq 1$. Ha így futtatjuk az algoritmust, akkor is egy $(1-\epsilon)^3$-közelítést kapunk az eredeti igényekre nézve. Az igények átskálázása után $\beta \leq k$.

Minden fázisban k iteráció történik. Ha egy iterációban p lépést hajt végre az algoritmus, akkor ebből az első $p-1$ olyan, hogy valamely él hosszát $(1+\epsilon)$-szorosára növeli. Nevezzük ezeket *teljes* lépéseknek. Egy él hossza kezdetben $\delta/c(e)$, az algoritmus futása után pedig kevesebb, mint $(1+\epsilon)/c(e)$, ugyanis a legutolsó növelés előtt $D < 1$ és így a súly kevesebb, mint $1/c(e)$. Eszerint egy él legfeljebb $\lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor$ teljes lépésben lehet a növelő út legkisebb kapacitású éle, azaz összesen legfeljebb $m \lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor =$ $= m \lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ teljes lépés lehet. A nem teljes lépések száma legfeljebb kt, így az eddigiek alapján az algoritmus összesen legfeljebb $(k^2 +$ $+ m) \lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1+\epsilon} \rceil$ lépést hajt végre.

Ha elég nagy az igények száma ($k \geq \sqrt{m}$), ez az eredmény megjavítható $(k \log k + m \log k) \lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1+\epsilon} \rceil$-ra a következő módon. Ha az algoritmus nem ér véget $T = \lceil \frac{2}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ fázis után, akkor tudjuk, hogy $\beta > 2$. Ebben az esetben szorozzunk be minden igényt 2-vel, így β-t megfelezzük, de $\beta \geq 1$ továbbra is fennáll, és kezdjük újra az algoritmust. Mivel a kezdeti átskálázás után $\beta \leq k$ volt, így ezt a menet közbeni átskálázást legfeljebb $\log k$-szor kell megcsinálni.

3.4. Minimális költségű törtfolyam irányítatlan gyűrűn

Ebben a fejezetben Sepherd és Zhang [3] munkája nyomán egy olyan eredményt tárgyalunk, amely irányítatlan gyűrűn minimális költségű folyamot ad. Ebben az esetben megengedett folyamot találni könnyű, a megengedett folyam létezésére a 4.3. fejezetben láthatunk feltételt és algoritmust. Az eredmény erőssége, hogy képes megtalálni a minimális költségű folyamot.

3.5. Definíció. *Legyen $G = (V, E)$ irányítatlan gráf, s_i, t_i, d_i termékek $(i = 1, ..., k)$. Minden termékhez adott egy $w_i : E \to R$ költségfüggvény. Az igényeket kielégítő $f_1, \ldots, f_k$ folyam költsége $\sum_{i=1}^{k} \sum_{e \in E} |f_i(e)| w_i(e)$.*

3.6. Definíció. *Egy n pontú irányítatlan gyűrűn a következő irányítatlan gráfot értjük: $G = (V, E) = (\{0, 1, \ldots, n-1\}, \{(0,1), (1,2), \ldots, (n-1, 0)\})$.*

MINIMÁLIS KÖLTSÉGŰ TÖRTFOLYAM IRÁNYÍTATLAN GYŰRŰN
Adott: n pontú irányítatlan gyűrű, c egész kapacitásfüggvény, s_i, t_i, d_i igények (d_i egész), minden termékhez és e élhez egy $w_i(e)$ költség.
Feladat: Adjuk meg az igényeket kielégítő minimális költségű megengedett többtermékes folyamot!

A továbbiakban erre a problémára adunk algoritmust. Először néhány jelölést vezetünk be.

3.4.1. Jelölések

A gyűrű pontjainak növő sorrendjét „előre” iránynak nevezzük. Azt mondjuk, hogy az $x_1, x_2, \ldots, x_l$ pontok előre sorrendben vannak, ha x_1-ből indulva előre irányban körüljárva a kört az adott sorrendben jönnek.

A továbbiakban feltesszük, hogy a gyűrű egy adott pontpárja között csak egy igény van. Így a költségfüggvény jelentősen egyszerűsödik, ugyanis egy igényt csak két úton szállíthatunk, előre, vagy hátra, ugyanakkor egy útnak egyértelmű a költsége, hiszen az adott úton csak azt az egyetlen terméket szállíthatjuk, amit az út végpontjai meghatároznak.

Jelölje $P(i,j)$ az $i, i+1, \ldots, j$ pontokból álló utat (modulo n). A $P(i,j)$ út költségét csak az i és j között menő termék költségfüggvényével érdemes venni. Ilyen termék csak egy van, így jelölje $w(ij) := w_l(P(i,j))$, ahol $s_l = i$ és $t_l = j$. A továbbiakban nem használjuk ki, hogy a $w(ij)$ költségeket élköltségekből kaptuk, azonban kihasználjuk a w egy egyszerű tulajdonságát:

3.7. Definíció. *Egy utakon értelmezett w költségfüggvény normális, ha minden i, j, k, l négyesre, amelyek előre sorrendben következnek a körön: $w(ij) + w(kl) \leq w(ji) + w(lk)$*

A megoldást egy $x : \mathcal{P} \to R^+$ vektor jelenti, amely esetünkben azt adja meg, hogy mennyit küldünk előre, illetve hátra a körön. Jelölje $x(ij) := x(P(i,j))$ az i és j között előre küldött mennyiséget, míg $x(ji)$ a hátra küldött mennyiséget.

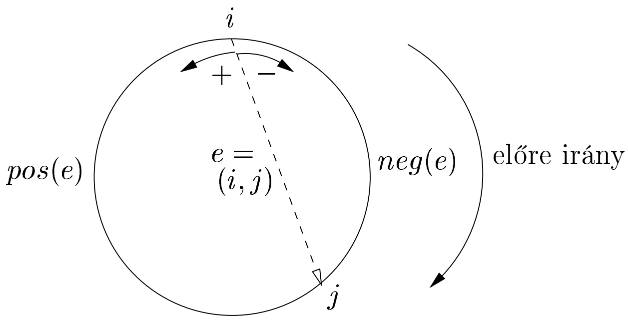
Egy e él *terhelése* $l(e) = \sum_{e \in P} x(P)$, az e él *maradék kapacitása* $c(e) - l(e)$. Az e él *telített*, ha maradék kapacitása 0. Az $s_i t_i$ igény *osztott*, ha $x(s_i t_i)$ és $x(t_i s_i)$ is pozitív.

A kör egy i-ből j-be menő irányított *húrját* (i,j)-vel jelöljük (4. ábra.) Legyen az x által meghatározott húrok halmaza $\mathcal{P}(x) := \{(i,j) : x(ij) > 0\}$. Az $e = (i,j)$ húr *minősége* $q(e) := w(ji) - w(ij)$. Amennyiben a húr minősége negatív, akkor olcsóbb a hátra úton $P(j,i)$-n szállítani a húr által meghatározott igényt.

A hátra utat $P(j,i)$ a húr *pozitív részének*, az előre utat $P(i,j)$ pedig a *negatív részének* nevezzük. Egy (i,j) húr *maximális*, ha nincs másik $(k,l) \in \mathcal{P}(x)$ amelyre $P(i,j) \subset P(k,l)$, jelölje a maximális húrok halmazát $\mathcal{M}(x)$. Az (i,j) és (k,l) húrok *nem keresztezők*, ha k és l is $\{i, i+1, \ldots, j\}$-ben vagy $\{j, j+1, \ldots, i\}$-ben van, egyébként *keresztezőnek* mondjuk őket. Ha e-vel jelöljük az (i,j) húrt, akkor $\overline{e}$-vel jelöljük a (j,i) húrt.

3.4.2. Az optimalitás feltétele

A lináris programozási dualitás tételből azonnal adódik az optimalitás következő feltétele:



4. ábra: Az $e = (i, j)$ húr, $pos(e)$ és $neg(e)$ elhelyezkedése.

3.8. Állítás. *Az x megengedett megoldás pontosan akkor optimális, ha létezik egy $z : E \to R^+$ függvény, amelyre*

- *ha $z(e) > 0$, akkor e szoros*
- *ha $x(ij) > 0$, akkor $w(ij) + z(ij) \le w(ji) + z(ji)$*

Bizonyítás. Tegyük fel, hogy minden pontpár között van igény, az i és j pontok között menő igény nagysága $d(ij)$, ami 0 is lehet. A feladat pontos megfogalmazása:

$$0 \le i < j \le n-1 \qquad x(ij) + x(ji) = d(ij) \tag{14}$$

$$e \in E \qquad \sum_{e \in P} x(P) \le c(e) \tag{15}$$

$$0 \le i < j \le n-1 \qquad x(ij) \ge 0 \tag{16}$$

$$\min \left(\sum_{P \in \mathcal{P}} w(P)x(P) \right) \tag{17}$$

ennek duálisa

$$0 \le i < j \le n-1 \qquad y(ij) - \sum_{e \in P(i,j)} z(e) \le w(P(i,j)) \tag{18}$$

$$y(ij) - \sum_{e \in P(j,i)} z(e) \le w(P(j,i)) \tag{19}$$

$$e \in E \qquad z(e) \ge 0 \tag{20}$$

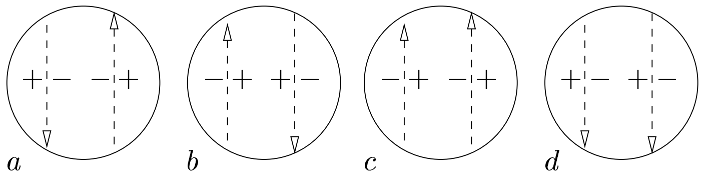
$$\max \left(\sum_{0 \le i < j \le n-1} y(ij)d(ij) - \sum_{e \in E} z(e)c(e) \right) \tag{21}$$

Az optimalitási feltételek a maximum és a minimum egyenlőségéből kiolvashatók. $\square$

3.4.3. Az algoritmus alapvető lépései

Legyen x egy megoldás, $e = (i, j)$ és $f = (k, l)$ két húr, $\epsilon > 0$. Jelölje $Swap(x, e, f, \epsilon)$ azt a folyamot, amelyben az $x(ij), x(kl)$ változókat csökkentjük ϵ-nal az $x(ji), x(lk)$ változókat pedig növeljük ϵ-nal. Legyen $pos(e, f) = pos(e) \cap pos(f)$, $neg(e, f) = neg(e) \cap neg(f)$. A módosítás akkor ad megengedett folyamot, ha $\epsilon \le x(ij), x(kl)$ és ha ϵ legfeljebb akkora, mint $pos(e, f)$-ben a legszorosabb él maradék kapacitásának fele. (Ugyanis $pos(e, f)$-ben 2ϵ-nal nő, $neg(e, f)$-ben 2ϵ-nal csökken, a többi élen pedig változatlan marad a terhelés.) Legyen tehát ϵ maximális, úgy, hogy a módosított folyam megengedett legyen, ekkor jelölje $Augment(x, e, f) := Swap(x, e, f, \epsilon)$.

Egy $e, f \in \mathcal{P}(x)$ nem keresztező pár pozitív és negatív részei az irányítástól függően négy féle módon tudnak elhelyezkedni (5. ábra.) Ha $pos(e, f)$ üres, akkor azt mondjuk, hogy e és f *ellentétes.* Amennyiben a két húr pozitív részei tartalmazzák egymást, akkor valamelyik húr nem maximális. Ha $neg(e, f)$ üres, akkor $q(e) + q(f) \ge 0$ a normalitás miatt. Ha e és f ellentétes, akkor $Augment(x, e, f)$ olyan megengedett folyamot ad, amelynek költsége nem nagyobb az eredetinél, hiszen w normális, és így $q(e) + q(f) \le 0$.



5. ábra: a: ellentétes pár, $q(e)+q(f) \le 0$; b: $q(e)+q(f) \ge 0$; c, d: valamelyik húr nem maximális

Az $e, f \in \mathcal{P}(x)$ keresztező pár *kifordított,* ha nincs szoros él $pos(e, f)$-ben. Egy *negatív pár* vagy egy olyan kifordított pár, amelyre $C(e, f) = q(e) + q(f)$ negatív, vagy egy e, e alakú „pár”, amelyre $pos(e)$-ben nincs szoros él, és $q(e) < 0$. (Ha ϵ-nal változtatjuk a folyamot, akkor a költség $\epsilon C(e, f)$-el változik.)

A fent definiált rengeteg jelölés nagyon egyszerű dolgokat szeretne megfogni. Egy $e, f \in \mathcal{P}(x)$ páron akkor tudjuk az $Augment(x, e, f)$ műveletet végrehajtani, ha a pozitív részben nincs szoros él. Amennyiben a pár nem keresztező, akkor az $Augment$ művelet akkor teszi olcsóbbá a folyamot, ha a pozitív rész üres, ekkor a művelet automatikusan végrehajtható. Az optimalitás nyilvánvaló akadályai a negatív párok, ugyanis a definíció szerint végrehajtható az $Augment$ művelet, mivel $pos(e, f)$-ben nincs szoros él, és a művelet utáni folyamköltség kisebb lesz, mivel $C(e, f)$ negatív. Ugyanez egyetlen húrra is elmondható.

3.4.4. Az algoritmus

Az algoritmus nagyvonalakban a következőképpen működik. Egy tetszőleges megengedett megoldásból indulunk ki, amelyet például az Okamura-Seymour tétel szolgáltathat (4.37. tétel). A `PACIFY` rutin eltünteti az ellentétes párokat, a `CLEANSE` rutin egy egész-fél megoldást szolgáltat, a főprogramban (`MCMRR`) ezt alakítjuk optimálissá. A rutinok leírása a következő:

`PACIFY(x)`

- `Legyen` x `megengedett megoldás`
- `Amíg van` (e, f) `ellentétes pár` $\mathcal{M}(x)$`-ben`
  - $x := Augment(x, e, f)$
  - `Az új` x`-nek megfelelően változtassuk meg` $\mathcal{P}(x)$`-et és` $\mathcal{M}(x)$`-et`

`CLEANSE(x)`

- `Legyen` x `egy megengedett megoldás, amelyben nincsenek ellentétes párok.`
- `Legyen` m `az osztott igények száma. Ha` $m = 0$`, akkor kész vagyunk.`
- **[Inicializálás]** `Az osztott igényeket írjuk fel a következő alakban:` $\{g_i = (v_i, v_{i+m})\}$`, ahol` $i = 0, ..., m-1$`, és a` $v_0, v_1, \ldots, v_{m-1}$ `pontok sorban jönnek a körön előre irányban végighaladva.`
- `Minden egyes` $i = 0, \ldots, m-1$ `osztott igényre`
  - `Válasszuk az` e, f `párt` $\overline{g_i}, g_{i+1}$`-nek vagy` $g_i, \overline{g_{i+1}}$`-nek úgy, hogy` $C(e, f) < 0$ `legyen.`
  - `Ha az` e, f `pár kifordított, akkor`
    - $x := Augment(x, e, f)$
    - `Ha` g_i `vagy` g_{i+1} `a módosítás után már nem osztott, akkor kezdjük az eljárást az` **[Inicializálás]**`-tól.`

`MCMRR(x)`

- `Legyen` x `megengedett megoldás`
- `PACIFY(x)`
- `CLEANSE(x)`

- `Amíg létezik` $e, f \in \mathcal{M}(x)$ `negatív pár`
  - $x := Augment(x, e, f)$
  - `Módosítsuk` $\mathcal{P}(x)$`-et és` $\mathcal{M}(x)$`-et, ha szükséges.`

3.4.5. Az algoritmus helyessége

Bár az algoritmus alapgondolata nagyon egyszerű – az optimalitás triviális akadályait elég eltüntetni – a helyesség bizonyítása nagyon sok technikai jellegű munkát igényel, így a bizonyításoknak csak a vázlatát írjuk le, a részletes bizonyítások [3]-ban találhatók.

Az algoritmus alapját képező *Augment* eljárást úgy definiáltuk (az ϵ választásával), hogy megengedett megoldásból megengedettet hoz létre, azaz az algoritmus futása során x mindig megengedett megoldás.

3.9. Állítás. *Ha nincs ellentétes pár $\mathcal{M}(x)$-ben, akkor nincs ellentétes pár $\mathcal{P}(x)$-ben sem.*

Bizonyítás. Tegyük fel, hogy e, f ellentétes, de legalább az egyik nem maximális. Ha az e húr nem maximális, akkor létezik egy e' húr, ami nála nagyobb. Ekkor az e' előre útja tartalmazza az e-hez tartozó előre utat, vagyis $pos(e') \subset pos(e)$. Az ellentéteség miatt $pos(e) \cap pos(f)$ üres, ekkor $pos(e') \cap pos(f')$ is üres, így e', f' is ellentétes, azaz $\mathcal{M}(x)$-ben is van ellentétes pár. □

3.10. Állítás. *Ha nincs ellentétes pár $\mathcal{M}(x)$-ben, akkor nincs ellentétes pár $Augment(x, e, f)$ után sem, amennyiben e, f maximális negatív pár.*

Bizonyítás. Tegyük fel, hogy a módosítás után $g = (i, j), h = (k, l)$ maximális ellentétes pár. Ezek a módosítás előtt nem voltak ellentétesek, ami csak amiatt lehet, hogy valamelyikük – tegyük fel, hogy g – nem volt $\mathcal{P}(x)$-ben, azaz $x(ij) = 0$ volt. A g húr csak úgy kerülhetett be $\mathcal{P}(x)$-be, ha $x(ij)$-t megnöveltük, így feltehető, hogy $e = (j, i)$. Mivel $P(j, i) \subset P(k, l)$, ezért $x(kl) = 0$ volt a módosítás előtt, különben e nem lett volna maximális, azaz $x(kl)$-t is megnöveltük, ami csak úgy lehet, ha $f = (l, k)$. Ekkor azonban $neg(e)$ és $neg(f)$ diszjunkt, vagyis e, f nem lehet negatív pár. □

Az állítás következményeképp a `CLEANSE` és `MCMRR` eljárások során nem keletkeznek ellentétes párok.

3.11. Állítás. *Tegyük fel, hogy $\mathcal{M}(x)$-ben nincsenek ellentétes párok.*

1. *Ha az e húr egy osztott igényhez tartozik, akkor e és $\overline{e}$ is $\mathcal{M}(x)$-beli.*
2. *Ha $e, f \in \mathcal{M}(x)$ és $q(e) + q(f) < 0$, akkor e és f keresztezőek.*

Bizonyítás. Nem bizonyítjuk. □

3.12. Tétel. *A* `CLEANSE` *rutin olyan x megoldást ad, amely*

- *fél-egész,*
- *minden él terhelése egész*
- *ha megnöveljük minden él kapacitását 1-el, akkor létezik egész megoldás, amelynek költsége legfeljebb x költsége.*

Bizonyítás. A bizonyítás vázlata: A 3.11. állítás első fele miatt az osztott igények valóban a `CLEANSE` rutin **[Inicializáció]** részében adott módon írhatók le. Megmutatható, hogy minden $P(v_i, v_{i+1})$ íven található egész terhelésű él, amiből elemi számolással adódik, hogy a megoldás fél-egész és minden él terhelése egész. Ezt a megoldást megfelelő módon kerekítve adódik a tétel harmadik része. □

Az algoritmus helyességét az alábbi tétel igazolja:

3.13. Tétel. *Az* `MCMRR` *algoritmus fél-egész és minimális költségű x megoldást ad.*

Bizonyítás. A bizonyítás vázlata: A 3.12. tétel szerint a `CLEANSE` által adott megoldás fél-egész, és minden él terhelése egész. Az algoritmus további részében az *Augment* lépésekben az ϵ választásakor csak fél-egész értékeket kaphatunk, így a megoldás fél-egész az élek terhelése pedig egész marad. A 3.10. állítás szerint a `PACIFY` végrehajtása után sosem keletkeznek ellentétes párok, így feltehető, hogy x-ben nincsenek ellentétes párok.

Az optimalitás bizonyításának ötlete az, hogy elkészítünk egy irányított, súlyozott segédgráfot, amelyben pontosan akkor van negatív kör, ha van az eredeti gráfban negatív pár. Ezek után ha az `MCMRR` rutin már nem talál negatív párt, akkor létezik a segédgráfban megengedett potenciál, ebből pedig megkonstruálható az optimalitást bizonyító z duális megoldás. □

Az algoritmus polinomiális idejű, azonban ennek megmutatására itt nem vállalkozunk.

4. Osztatlan folyamok

A 2.7. fejezetben definiált osztatlan folyam problémában adott egy irányított gráfunk, és adottak igények. A feladatunk az, hogy minden igényhez találjunk egy útvonalat, amelyen az adott igényt kielégítjük, úgy, hogy az élkapacitásokat ne sértsük.

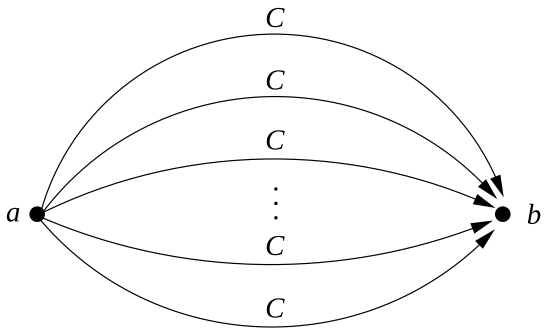
Ebben a fejezetben különböző approximációs algoritmusokat ismertetünk, amelyek speciális eseteket oldanak meg: vagy a gráf topológiája speciális, vagy az igények elrendezése, vagy mindkettő. Végül egy ütemezéselméleti alkalmazását ismertetjük az osztatlan folyam problémának.

4.1. A feladat bonyolultsága

4.1. Tétel. *Általános gráfban, tetszőleges igényekre annak megállapítása, hogy létezik-e osztatlan folyam NP-nehéz.*

Bizonyítás. Visszavezetjük az NP-nehéz hátizsák pakolási feladatot az osztatlan folyam problémára. A hátizsák pakolási feladatban adott m darab tároló, amelyek mérete C, és adott k darab tárgy, amelyek mérete rendre $d_1, \ldots, d_k$. A kérdés az, hogy el tudjuk-e helyezni a tárgyakat a tárolókban úgy, hogy minden tárolóban a benn lévő tárgyak méretének összege legfeljebb C.

Tegyük fel, hogy adott $m, C, k, d_1, \ldots, d_k$, ekkor készítsük el a 6. ábrán látható gráfot. Vegyünk fel k darab igényt, amelyek forrása az s csúcs, nyelője pedig a t csúcs. Az i. igény mérete legyen d_i.



6. ábra: A hátizsák pakolási feladatnak megfelelő gráf

Ha létezik megoldása az így definiált osztatlan folyam problémának, akkor az i. tárgyat tegyük abba a tárolóba, ahányadik élen át megy az i. termék a-ból b-be. A folyam megengedettsége pontosan azt jelenti, hogy egyik tárolóba sem kerül C-nél több összméretű tárgy. Megfordítva, a pakolási feladat egy megoldásából elkészíthetjük az osztatlan folyam probléma egy megoldását úgy, hogy ha egy tárgy az n. tarolóba került, akkor a neki megfelelő terméket az n. élen át visszük a-ból b-be. □

4.2. Minimális terhelés irányított gyűrűn

4.2. Definíció. *Irányított gyűrű alatt a következő gráfot értjük:* $V = \{0, \ldots, n-1\}$, $E = \{(0,1), (1,0), (1,2), (2,1), \ldots, (n-2, n-1), (n-1, n-2), (n-1, 0), (0, n-1)\}$.

Egy irányított gyűrű tehát egy körgyűrű, amelynek szomszédos pontjai között mindkét irányba megy egy-egy él. A csúcsok $0, 1, \ldots, n-1$ sorrendjét *előre iránynak* nevezzük, a fordított sorrendet pedig *hátra iránynak*. A jelölések könnyítése végett ebben az alfejezetben a csúcsokkal végzett műveletek modulo n értendők. Tehát a gráf élei $(i, i+1)$ és $(i+1, i)$ alakúak $i = 1, ..., n-1$. Itt $(i, i+1)$ *előre él*, míg $(i+1, i)$ *hátra él*.

4.3. Definíció. *$\overrightarrow{i,j}$ íven az $i, i+1, i+2, \ldots, j$ pontokat értjük (modulo n). Az $u_1, u_2, ..., u_l$ pontok sorban vannak, ha u_1-ből kiindulva előre irányban körbejárva a kört az adott sorrendben következnek.*

Ezzel a jelöléssel az i pontból a j pontba két út létezik, az $\overrightarrow{i,j}$ *előre út*, és a $\overrightarrow{j,i}$ *hátra út*. Az igényeket legkönnyebben a kör átlóiként szemléltethetjük. Minden igény esetén két választásunk van: vagy előre irányítva elégítjük ki, vagy hátra irányítva. Pontosan megfogalmazva a következő feladatot szeretnénk megoldani:

MINIMÁLIS TERHELÉS IRÁNYÍTOTT GYŰRŰN
Adott: A gyűrű csúcsainak száma n, k darab igény $0 \leq s_i, t_i \leq n-1$ alakban. (Ugyanarra a pontpárra több igény is meg lehet adva.)
Feladat: Adjuk meg azt a minimális L számot, hogy ha a gyűrű minden élének kapacitása L, akkor létezik az igényeket kielégítő osztatlan folyam (az igényeket 1 nagyságúnak véve), és adjunk meg egy ilyen folyamot.

A feladat azért érdekes ebben a formában, mert távközlési alkalmazásokban optikai hálózatok tervezésekor széles körben használt a gyűrű topológia. Egy gyűrű esetén előre ismert, hogy az egyes pontpárok között mennyi kommunikációs igény van, és ennek megfelelően kell beállítani a gyűrű éleinek kapacitását. Ez a gyakorlatban a gyűrű minden élére ugyanaz az L szám. Ezentúl azt is meg kell mondani, hogy egy igény előre, vagy hátra menjen a gyűrűn. A feladatot azért osztatlan folyam modellezi, mert van egy egységnyi információmennyiség, amely nem osztható több részre. Az alábbiakban Wilfong és Winkler algoritmusát ismertetjük [5].

4.2.1. Az algoritmus és helyessége

Tekintsük először a probléma lineáris programozási relaxációját. Jelölje $x_i \in [0, 1]$, hogy az i. igényből mennyit küldünk előre, ekkor persze $1 - x_i$-t hátra felé küldünk a gyűrűn. Jelölje az $(l, l+1)$ él *terhelését*, azaz az élen áthaladó folyamösszeget A_l, az $(l+1, l)$ él terhelését pedig B_l:

$$A_l = \sum_{l \in [s_i, t_i - 1]} x_i \qquad B_l = \sum_{l \in [t_i, s_i - 1]} (1 - x_i) \tag{22}$$

Az x irányítás által indukált *terhelést*, azaz a maximális élterhelést jelölje L:

$$L = \max\left(\max_l A_l, \max_l B_l\right)$$

L-et tudjuk minimalizálni polinom időben, mivel ez egy lineáris program. Az optimumot jelölje L^*, az ezt megvalósító irányítást pedig x^*. Az eredményben x_i^* tetszőleges $[0,1]$-beli valós szám lehet. Azt mondjuk, hogy az i. igény *osztott*, ha a hozzá tartozó változó nem 0 vagy 1 értékű.

Az eredeti feladat szerint azonban olyan x vektort keresünk, amelyre $x_i \in \{0,1\}$ és L minimális. Legyen az eredeti feladat optimális megoldása L_{OPT}. Szükségünk lesz még a következő fogalomra is:

4.4. Definíció. *Egy x megoldás egészösszegű, ha $\sum_{i=1}^{k} x_i$ egész.*

Tekintsük most az egészösszegű x vektorokat, az ezekkel elérhető minimális terhelés értéket jelölje L'. Az az egészösszegű vektor, ahol ez fölvétetik legyen x'. Mivel egy 0-1 értékű vektor egészösszegű, az egészösszegűség pedig megszorítása a relaxációnak:

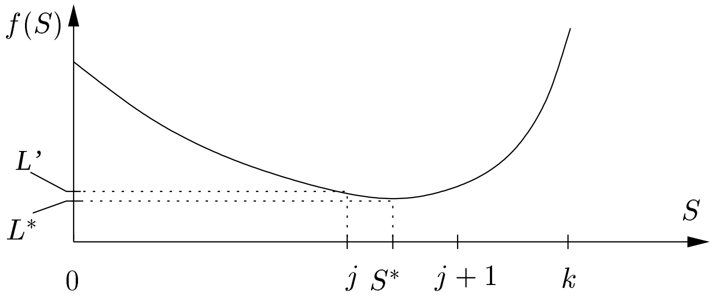
$$L^* \le L' \le l_{OPT}.$$

4.5. Lemma. *Az L' értéket polinom időben meg tudjuk határozni. $L' <$ $< L_{OPT} + 1$. [5]*

Bizonyítás. L' meghatározásának egyik módja lehet, hogy a lineáris programhoz hozzáírjuk a $\sum_{i=1}^{k} x_i = S$ feltételt, és minden szóbajövő S értékre (S=$0, ..., k$) lefuttatjuk a lineáris programot. Az eredményül adódó terhelés értékek közül a minimális lesz az L'.

Szerencsére nincs szükség arra, hogy ezt a $k+1$ darab lineáris programot megoldjuk. Jelölje ugyanis $f(S)$ a minimális terhelést S változóösszeg mellett. Az f függvény konvex, ugyanis ha az $f(S_1)$ optimális terhelés az x^1 vektorral vétetik fel, amelynek összege S_1, az $f(S_2)$ optimális terhelés pedig az x^2 vektorral vétetik fel, aminek összege S_2, akkor $S = \alpha S_1 + (1-\alpha)S_2$-re az $x := \alpha x^1 + (1-\alpha)x^2$ olyan vektor, aminek összege S, terhelése pedig legfeljebb $\alpha f(S_1) + (1-\alpha) f(S_2)$. Ennél $f(S)$ nem lehet nagyobb, hiszen $f(S)$ az optimális terhelés értéke S változóösszeg mellett, azaz f valóban konvex. Tegyük fel, hogy az L^* relaxált optimum az S^* változóösszeg mellett vétetik fel ($f(S^*) = L^*$). Ekkor az f egész helyeken vett minimumát csak az S^*-hoz legközelebbi két egészen (j és $j+1$) veheti fel (7. ábra). Így elég erre a kettőre megkötni a változóösszeget és futtatni a lineáris programot.

Az hogy L' kevesebb, mint 1-el nagyobb L^*-nál azonnal következik abból, hogy x^*-ból kiindulva kaphatunk egy egészösszegű megoldást úgy, hogy addig növelünk változókat, amíg a változóösszeg egész nem lesz. Ekkor összesen kevesebb, mint 1-el változtattuk meg a változókat, így valamennyi A_l és B_l kevesebb, mint 1-el változik, tehát a terhelés L^*-hoz képest kevesebb, mint 1-el nőhet. Az így kapott terhelésnél L' nem nagyobb. □



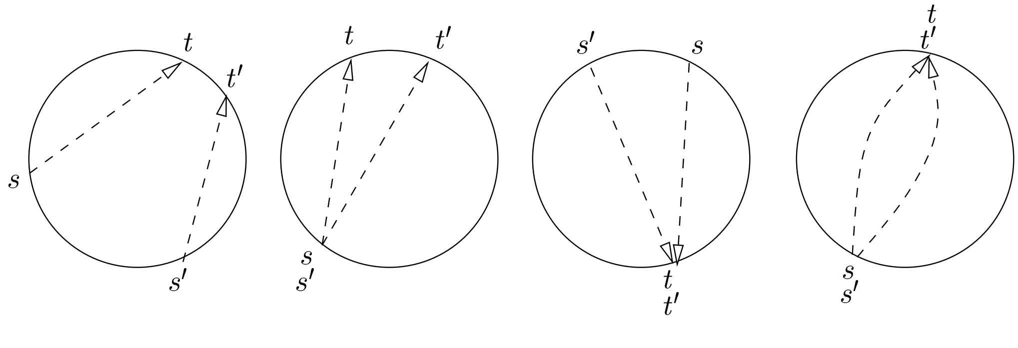
7. ábra: Az f függvény

4.6. Tétel. *A* Minimális terhelés irányított gyűrűn *feladat megoldható polinom időben. [5]*

Bizonyítás. Feladatunk tehát az, hogy L_{OPT} értékét meghatározzuk. A lineáris programozási feladatot meg tudjuk oldani, ekkor megkapjuk L^* értékét, és egy x^* irányítást, amelynek összege $\sum_{i=1}^{k} x_i^* = S^*$. Ezután meghatározunk egy x' egészösszegű optimális irányítást, az 4.5. lemmában bemutatott módon. Ekkor tehát $\sum_{i=1}^{k} x_i'$ egész, és az x'-hez tarozó L' terhelésre $L' \leq L_{OPT}$.

4.7. Definíció. *Azt mondjuk, hogy az s, t és az s', t' igények párhuzamosak, ha s-ből kiindulva előre irányban végigjárva a kör pontjait, a két igény kezdő és végpontjai s, s', t', t vagy s, t, t', s' sorrendben következnek. (Ezen pontok közül néhányan egybe is eshetnek.)*

Ha az igényeket a kör átlóiként képzeljük el, akkor ez azt jelenti, hogy az igények nem metszik egymást, és „egyirányúak”. Egyszerű észrevétel, hogy ha két igény kezdő, vagy végpontja megegyezik, akkor azok párhuzamosak. Párhuzamos igényekre a 8. ábrán láthatunk példákat.



8. ábra: Párhuzamos igények

Az x' irányítás mellett jelölje az egyes éleken a terhelést (22)-nek megfelelően A_k' és B_k'. A következő lépésében párhuzamos osztott igénypárokkal fogunk foglalkozni. Legyenek tehát az s_i, t_i és s_j, t_j párhuzamos igények osztottak, azaz $0 < x_i, x_j < 1$. A következő egyszerű módosítást *kikeresztezésnek* nevezzük:

1. Ha $1 - x_j \leq x_i$, akkor legyen $\epsilon = 1 - x_j$, növeljük x_j-t ϵ-nal és csökkentsük x_i-t ϵ-nal. A változóösszeg nem változik, és könnyen ellenőrizhető, hogy semelyik élen sem nő a terhelés, sőt néhány élen csökken. A változtatás után $x_j = 1$ fog fennálni ($x_i \geq 0$ is teljesül), azaz eggyel kevesebb osztott igény lesz.

2. Amennyiben $1 - x_j > x_i$, akkor legyen $\epsilon = x_i$, csökkentsük x_i-t ϵ-nal, x_j-t pedig növeljük ϵ-nal. Ugyanúgy igaz lesz, hogy az élek terhelése nem nő, a változtatás után $x_i = 0$ lesz, azaz az osztott igények száma ebben az esetben is csökken.

Mivel egy kikeresztezés során az osztott igények száma eggyel csökken, ezért legfeljebb k kikeresztezési lépés után olyan helyzet áll elő, hogy az osztott igények között már nem lesznek párhuzamos párok. Ekkor a kör minden pontjából legfeljebb egy osztott igény indul ki, hiszen ha lenne két osztott igény, amelynek ugyanaz a kezdőpontja, akkor ezek párhuzamosak lennének, ami nem lehet, hiszen a kikeresztezések után nincsenek osztott párhuzamos párok. Ez azt is jelenti, hogy az eljárás után az osztott igények száma legfeljebb n. A kikeresztezések során az élek terhelése nem nő.

A továbbiakban csak az osztott igényekkel fogunk foglalkozni. Legyen az osztott igények száma $q \leq n$, feltehető, hogy az igények úgy vannak számozva, hogy az osztott igények halmaza: $(s_1, t_1), \ldots, (s_q, t_q)$, továbbá a körön előre irányban végigmenve az $s_1, \ldots, s_q$ pontok sorban vannak.

4.8. Állítás. *Ekkor a $t_1, \ldots, t_q$ pontok is sorban vannak.*

Bizonyítás. Ha ez nem így van, akkor feltehető, hogy a 2. és 3. igény van fordítva, azaz a körön előrehaladva az első három nyelő t_1, t_3, t_2 sorrendben következik. Mivel az osztott igények közt nincs párhuzamos, ezért a párhuzamosság definíciója miatt t_2, vagy t_3 s_2 és s_3 közé esik.

1. Ha t_3 és t_2 is s_2 és s_3 közé esik, akkor ennek a 4 pontnak és s_1-nek a sorrendje s_1, s_2, t_3, t_2, s_3. Ekkor t_1 nem lehet az $\overrightarrow{s_1, t_3}$ íven, mert akkor az s_3, s_1, t_1, t_3 sorrend miatt a 3. és 2. igény párhuzamos lenne. De nem lehet a $\overrightarrow{t_2, s_1}$ íven sem, mert ekkor a s_1, s_2, t_2, t_1 sorrend miatt az 1. és 2. igény lenne párhuzamos.

2. Ha csak t_3 esik s_2 és s_3 közé, akkor az előbbi öt pont sorrendje: s_1, s_2, t_3, s_3, t_2. Ekkor ha t_1 az $\overrightarrow{s_1, t_3}$ íven van, akkor az 1. és 3. igény párhuzamos, ha pedig a $\overrightarrow{t_2, s_1}$ íven, akkor az 1. és a 2. igény lesz párhuzamos.

3. Az az eset, mikor csak t_2 esik s_2 és s_3 közé, hasonlóan kezelhető.

□

Ha egy élet igényként képzelünk el, akkor értelmezni tudjuk egy él és egy igény párhuzamosságát is. Egyszerű észrevétel, hogy az x_i változó megváltoztatása pontosan azon élek terhelését változtatja meg, amelyek párhuzamosak s_i, t_i-vel. Melyek azok az igények, amelyek egy adott $(l, l+1)$ éllel párhuzamosak? A 4.8. állítás miatt ezek a $\{1, 2, \ldots, q\}$ halmaz egy $[a_l, b_l]$ intervallumát adják, ahol az intervallumot modulo q értjük. Az $(l+1, l)$ éllel párhuzamos igények pedig pontosan az előző intervallum komplementerében vannak.

Az algoritmus utolsó lépésében kerekítjük a kikeresztezés után még osztott igényeket. Adott tehát az x' egészösszegű irányítás a kikeresztezés után, ekkor rekurzívan definiáljuk x-et:

$$x_j := \begin{cases} 1 & \text{ha } -x'_j + \sum_{i=1}^{j-1}(x_i - x'_i) < -\frac{1}{2}, \\ 0 & \text{különben.} \end{cases} \tag{23}$$

Ez a definíció azt garantálja, hogy minden $j \leq q$-ra

$$\sum_{i=1}^{j}(x_i - x'_i) \in \left[-\frac{1}{2}, \frac{1}{2}\right). \tag{24}$$

Jelölje az így kapott x irányításból adódó élterheléseket A_k és B_k. Mivel csak az első q darab igényhez tartozó változót változtattuk meg, ezért:

$$A_k = A'_k + \sum_{i \in [a_k, b_k]}(x_i - x'_i) \qquad B_k = B'_k + \sum_{i \notin [a_k, b_k]}(x'_i - x_i)$$

4.9. Állítás. *Bármely* $1 \leq a, b \leq q$*-ra* $\sum_{i \in [a,b]}(x_i - x'_i) < 1$.

Bizonyítás. Ha $a < b$, akkor (24)-et felhasználva

$$\sum_{i \in [a,b]}(x_i - x'_i) = \sum_{i=1}^{b}(x_i - x'_i) - \sum_{i=1}^{a-1}(x_i - x'_i) < \frac{1}{2} - \left(-\frac{1}{2}\right) = 1.$$

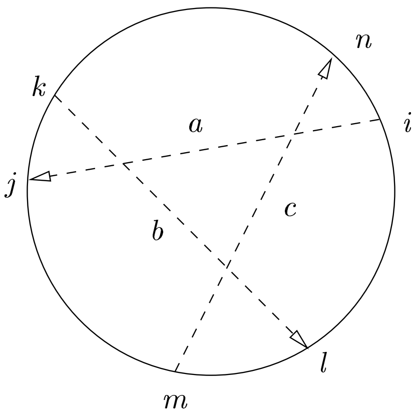
Ha $a > b$, akkor

$$\sum_{i \in [a,b]}(x_i - x'_i) = \sum_{i=1}^{q}(x_i - x'_i) - \sum_{i=1}^{b}(x_i - x'_i) + \sum_{i=1}^{a-1}(x_i - x'_i)$$

Tudjuk, hogy $\sum_{i=1}^{q}(x_i, x'_i) \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, de azt is tudjuk, hogy x' egész összegű, x szintén egész összegű – sőt egész is – így az első szumma $\left[-\frac{1}{2}, \frac{1}{2}\right)$-beli egész, vagyis 0. A második és harmadik tagot (24) alapján becsülve az állítás adódik. □

Ugyanez a gondolat hasonlóképpen elmondható a hátra élek esetén is, így $B_k - B'_k < 1$ is igaz. Ebből az következik, hogy $L < L' + 1 \leq L_{OPT} + 1$. Mivel L_{OPT} és L egész, ezért ez azt jelenti, hogy $L_{OPT} = L$, azaz a kapott x vektor által meghatározott irányítás optimális. Ezzel a tétel bizonyítását befejeztük. $\square$

A kikeresztezési és kerekítési eljárás során az élek terhelése nem nő. Ha $L^* < L_{OPT}$, akkor annak csak az lehet az oka, hogy létezik olyan példa, amiben van törtmegoldás, de nincs egészösszegű megoldás. Egy ilyet láthatunk a 9. ábrán. Ha minden él kapacitását 1-re állítjuk, akkor az igények felét előre, másik felét hátra küldve megengedett törtmegoldást kapunk ($x_a = 1/2, \quad x_b = 1/2, \quad x_c = 1/2$). Mivel bármely két igényhez tartozó előre útnak van közös része, ezért $x_a + x_b \leq 1$, $x_b + x_c \leq 1$, $x_c + x_a \leq 1$, azaz $x_a + x_b + x_c \leq \frac{3}{2}$. Ugyanígy bármely két igényhez tartozó hátra útnak is van közös része, vagyis $x_a + x_b \geq 1$, $x_b + x_c \geq 1$, $x_c + x_a \geq 1$ és így $x_a + x_b + x_c \geq \frac{3}{2}$. Egészösszegű megoldás tehát nem létezhet.



9. ábra: Van törtmegoldás, de nincs egészösszegű megoldás.

4.3. Az Okamura-Seymour tétel és irányított változata

A következőkben az élidegen út problémával foglalkozunk. Az élidegen út probléma olyan osztatlan folyam probléma, amelyben minden igény, és minden kapacitás 1 nagyságú. (Tetszőleges kapacitásfüggvényre is igazak az eredmények, hiszen – mivel az igények 1 nagyságúak – egy c kapacitású élet kicserélhetünk c darab 1 kapacitásúra. Az algoritmusokban persze vigyáznunk kell, hogy a futási idő független legyen a kapacitások konkrét értékétől!)

Élidegen út probléma
Adott: $G = (V, E)$ irányított (irányítatlan) gráf, $s_i, t_i \in V$ $(i = 1, ..., k)$.
Feladat: Adjunk minden igényre egy $s_i \to t_i$-utat úgy, hogy a különböző igényekhez rendelt utak élhalmaza diszjunkt legyen.

Az igények alapján definiálhatjuk a $H = (V, I)$ úgynevezett igénygráfot is, a H élei a (t_i, s_i) élek. Ezzel a jelöléssel a feladat úgy is fogalmazható, hogy a H éleit tartalmazó diszjunkt köröket keresünk a $G + H = (V, E \cup I)$ gráfban.

A probléma már két út esetén is NP-teljes, azonban ismertek speciális esetek, amikor a probléma polinomiális időben kezelhető. Ezekben az esetekben többnyire a G vagy a $G+H$ gráf síkbarajzolhatóságát vagy Euler-ségét, illetve az igények speciális elhelyezkedését követelik meg (egyéb feltételek mellett). Ezek az eredmények a probléma irányítatlan esetére vonatkoznak.

Az alábbiakban Frank András, Dorothea Wagner és Karsten Weihe publikálatlan eredményét ismertetjük, amely egy speciális esetben szükséges és elégséges feltételt ad a megoldás létezésére. A bizonyítás algoritmikus, az algoritmus vagy elkészíti a megoldást, vagy bizonyítékot szolgáltat arra, hogy nem létezhet megoldás. Az általunk tárgyalt esetben G síkbarajzolható irányított gráf, a $G + H$ Euler, az igényélek a G gráf egyetlen tartományának pontjait kötik össze. Feltesszük még, hogy a gráfban nincs jobb-kör, és hogy az igényélek egy referenciaponthoz képest „balról jobbra mutatnak". Ezen feltételek mellett a vágásfeltétel szükséges és elégséges a feladat megoldásához. Ebből az eredményből levezetjük Okamura és Seymour híres tételét [7].

4.3.1. Jelölések

4.10. Definíció. *A G síkgráf duálisán azt a G^* síkgráfot értjük, amely pontjai G tartományai, élei pedig szomszédos tartományokat kötnek össze. Jelölje G külső tartományát t_0.*

A 1. táblázatban összefoglaljuk a gráfbeli alakzatok duális megfelelőjét. Mivel a duális gráf duálisa az eredeti gráf, ezért a megfeleltetés egy-egyértelmű. A megfeleltetéseknek megfelelő állításokat *nem* bizonyítjuk. A pontos definíciókat és a velük kapcsolatos tételeket lásd [6]-ban.

Az eredeti G gráf irányításának megfelelően megirányítjuk a G^* éleit is úgy, hogy irányított vágásnak irányított kör feleljen meg. Ha az $e = (uv)$ élet irányított értelemben értjük, akkor $\overrightarrow{e}$-t vagy $\overrightarrow{(uv)}$-t írunk. Ugyanígy a $\overrightarrow{P}$ irányított utat jelöl.

Azt mondjuk, hogy a $\overrightarrow{C}$ irányított kör *jobb-kör*, ha a neki megfelelő C^* vágás t_0-t nem tartalmazó oldalának befoka 0. Ennek megfelelően az eredeti és a duális gráf új kapcsolatai jönnek létre, amelyeket a 2. táblázat foglal össze. (Ezeket szintén nem bizonyítjuk.) Irányított vágáson olyan halmazt értünk, amelynek befoka vagy kifoka 0.

A v pontra illeszkedő éleken ciklikus rendezést definiálhatunk a következő módon: irányítsuk meg ezen éleket a v pontból kifelé, ekkor $\{v\}$ egy 0 befokú irányított vágás lesz, aminek a duálisban megfelel egy irányított kör. Ezen a körön előre irányban végighaladva a duális élek és így az eredeti élek

G eredeti gráf	G^* duális gráf
e él	e két oldalán lévő tartományokat összekötő e^* él
t tartomány	t pont
v pont	v tartomány
C kör	C^* vágás (a duális pontok egy részhalmaza által meghatározott vágás)
ponthalmaz által meghatározott vágás élhalmaza: S	S^* kör
t tartomány határoló pontjai	t pontot tartalmazó tartományok
v pontot tartalmazó tartományok	v tartomány határoló pontjai
C kör belsejében lévő tartományok	t_0-ból nem elérhető pontok

1. táblázat: Irányítatlan struktúrák kapcsolata a dualitással.

G eredeti gráf	G^* duális gráf
$\overrightarrow{C}$ irányított kör	C^* irányított vágás
S irányított vágás	$\overrightarrow{S^*}$ irányított kör
$\overrightarrow{C}$ jobb-kör	C^* irányított vágás, amelynek befoka 0.

2. táblázat: Irányított struktúrák kapcsolata a dualitással.

egy ciklikus sorrendjét kapjuk, amelyet az élek *bal-sorrendjének* nevezünk. A definíció szerint az $\overrightarrow{(u,v)}$ él duálisa az u-ra illeszkedő élek duálisai által alkotott körben a bal-sorrendnek megfelelő irányba megy. A definíció szemléletes jelentése az, hogy ha az eredeti gráf jobb-körei az óramutató járásával megegyező irányba haladnak a lerajzolás szerint, akkor egy pontra illeszkedő élek bal-sorrendje az óramutató járásával ellentétes irányú.

A továbbiakban az ábrákon a duális éleket pontozott vonallal, a gráf éleket pedig folytonos vonallal jelöljük. Amennyiben egy élen nincs nyíl, akkor az él iránya az ábra szempontjából lényegtelen.

4.3.2. Az algoritmusban használt alapvető lépések

Az algoritmus szempontjából praktikus a G gráfot kissé átalakítanunk a következőképpen: vegyünk fel egy v' pontot a t_0 külső tartomány belsejében, és tegyük fel, hogy v' össze van kötve néhány t_0 határán lévő ponttal. (Az összekötés módját később határozzuk meg.) Az így kapott G' gráf is síkgráf. A G' gráf *külső tartományain* a v' pontra illeszkedő tartományokat értjük. Ezen tartományok uniója adja az eredeti t_0 tartományt.

Az algoritmus futása során sokszor lesz szükség speciális utak keresésére. Ezek tulajdonságait vizsgáljuk meg most.

4.11. Definíció. *Az $f = (vw)$ él az $e = (uv)$ éltől legjobbra (legbalra) lévő él az E élhalmazra nézve, ha a v pontra illeszkedő élek bal-sorrendjében f az e után következő első (utolsó) E-beli él.*

4.12. Definíció. *A $\overrightarrow{P} = \{\overrightarrow{e_1} = \overrightarrow{(v', p)}, \overrightarrow{e_2}, \ldots, \overrightarrow{e_l} = \overrightarrow{(q, v')}\}$ irányított körséta p-ből q-ba menő v'-út, ha csak egyszer halad át v'-n. A $\overrightarrow{P}$ belsején az $\{\overrightarrow{e_2}, \ldots, \overrightarrow{e_{l-1}}\}$ utat értjük.*

4.13. Definíció. *Azt mondjuk, hogy a $\overrightarrow{P} = \{e_1 = (v', p), e_2, \ldots, e_l = = (q, v')\}$ v'-út p-ből q-ba menő leg-jobb (leg-bal) út az E élhalmazra nézve, ha minden $i = 1, \ldots, l-1$-re e_{i+1} az e_i-hez képest legjobbra (legbalra) lévő v-ből kimenő él E-ben*

4.14. Definíció. *Jelölje az e él megfordítottját $\overline{e}$, az E éleinek megfordításával keletkező élhalmazt pedig $\overline{E}$. Egy $\overrightarrow{P} = \{e_1 = (v', p), e_2, \ldots, e_l = = (q, v')\}$ körséta leg-jobb (leg-bal) vissza-út az E élhalmazra nézve, ha $\overline{P} := \{\overline{e_l}, \overline{e_{l-1}}, \ldots, \overline{e_2}, \overline{e_1}\}$ leg-jobb (leg-bal) út $\overline{E}$-re nézve.*

A fenti definíciókban azért mondjuk egy körsétára, hogy út, mivel az a fontos, hogy egy ilyen körséta első és utolsó élét elhagyva a t_0 két pontját összekötő irányított utat kapunk. A körré való kiegészítés csak technikai szempontból szükséges.

Egyszerű észrevétel, hogy – amennyiben létezik – egy leg-jobb (leg-bal) út élszámban lineáris időben megtalálható, ha adott az út második $p \in t_0$ pontja. Legyen ugyanis $e_1 = (v', p)$, a továbbiakban pedig a legutoljára kiválasztott éltől legjobbra (legbalra) lévő még nem használt kimenő élet kell választani az út következő élének. Ha visszaérünk a v' pontba, akkor találtunk egy leg-jobb (leg-bal) utat, különben az eljárás elakad, és az aktuális végpontból már csak olyan élek indulnak ki, amelyeket az út már felhasznált. Ebben az esetben nem létezik (v', p)-vel kezdődő leg-jobb (leg-bal) út. A leg-jobb (leg-bal) vissza út is ugyanígy megtalálható, ha adott a keresett út utolsó előtti $q \in t_0$ pontja.

Jelölje a külső tartományt (t_0-t) határoló kört $C = \{v_0, v_1, \ldots, v_{l-1}\}$, ahol a pontokat úgy soroltuk fel, hogy ha a kör éleit $\overrightarrow{(v_i, v_{i+1})}$ irányban irányítjuk meg, akkor jobb-kört kapunk. (Feltehető, hogy a G gráf kétszeresen pont-összefüggő, ebben az esetben a tartományok határa tényleg egy kör.) Jelölje $P(v_i, v_j)$ a $\{v_i, v_{i+1}, \ldots, v_j\}$ utat.

4.15. Definíció. *Az $\overrightarrow{e}$ él jobb oldalán (bal oldalán) lévő tartományon az $\overrightarrow{e}^*$ duális él kezdőpontjának (végpontjának) megfelelő tartományt nevezzük.*

4.16. Definíció. *Egy t tartomány elérhető a D G-beli körsétára nézve, ha a duálisban létezik irányítatlan értelemben vett út t_0-ból t-be amely nem használja D éleinek duálisát.*

4.17. Állítás. *A $\overrightarrow{C}$ irányított G-beli kör pontosan akkor jobb-kör, ha valamely élének baloldala elérhető $\overrightarrow{C}$-re nézve.*

Bizonyítás. Ha $\overrightarrow{C}$ irányított kör, akkor a C^* irányított vágás bármely $\overrightarrow{e^*} = \overrightarrow{(t,t')}$ élére t és t' közül pontosan az egyik érhető el t_0-ból.

Ha $\overrightarrow{C}$ jobb-kör, akkor t' (az e él bal oldala) van a vágás t_0-t tartalmazó oldalán.

Ha a kör egy élének $\overrightarrow{e^*} = \overrightarrow{(t,t')}$ duálisára t' elérhető t_0-ból, akkor e^* a duális vágásból kifelé mutat. Mivel C^* irányított vágás, ezért nem lehet befelé mutató él, így $\overrightarrow{C}$ jobb-kör. □

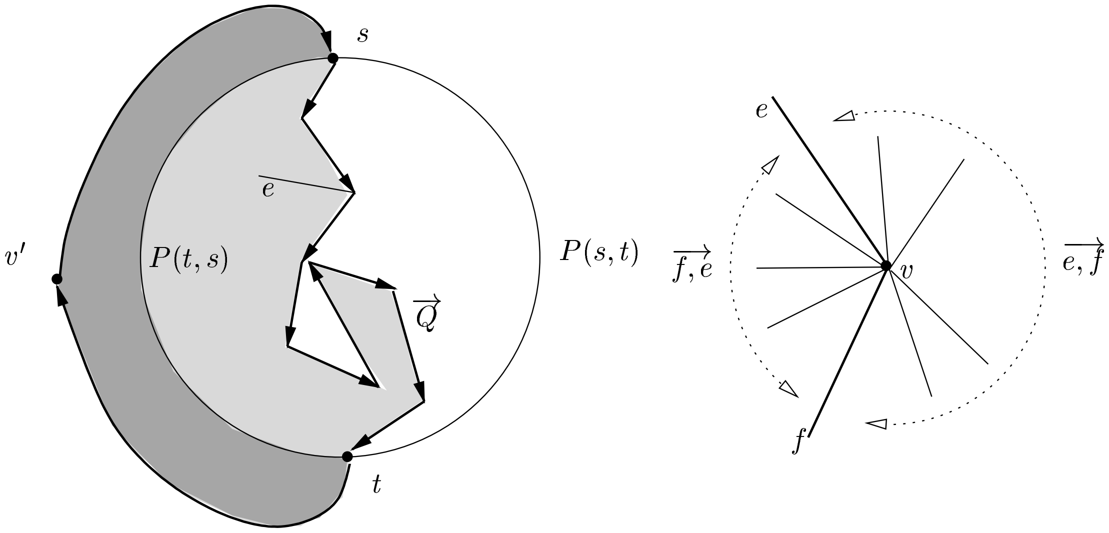
4.18. Definíció. *Ha $\overrightarrow{Q} = \{e_1 = (v', s), e_2, \ldots, e_l = (t, v')\}$ v'-út, akkor legyen $T'(\overrightarrow{Q})$ azon v'-re illeszkedő tartományok halmaza, amelyeknek van éle a $P(t,s)$ útról. Legyen $T(\overrightarrow{Q})$ azon tartományok halmaza, amik elérhetők $T'(\overrightarrow{Q})$-beli tartományból a $\overrightarrow{Q}$ körsétára nézve.*

4.19. Definíció. *A $\overrightarrow{Q}$ v'-út jobb oldalán a következő $j(\overrightarrow{Q})$ élhalmazt értjük: $\overrightarrow{e} = \overrightarrow{(u,v)} \in j(\overrightarrow{Q})$ pontosan akkor, ha u vagy v eleme $\overrightarrow{Q}$-nak, és $\overrightarrow{(u,v)}$ mindkét oldala $T(\overrightarrow{Q})$-beli tartomány. Ha $u \in \overrightarrow{Q}$, akkor azt mondjuk, hogy $\overrightarrow{e}$ a $\overrightarrow{Q}$ jobb oldalából kimenő él, ha $v \in \overrightarrow{Q}$, akkor a jobb oldalba bemenő élnek mondjuk. (E két eset nem zárja ki egymást.)*

4.20. Definíció. *A v-re illeszkedő e, f élek által meghatározott $\overrightarrow{e,f}$ él-intervallumon azon v-re illeszkedő éleket értjük, amelyek a bal-sorrend szerinti körüljárás esetén f után, de e előtt következnek.*

A fenti három definícióban szereplő fogalmak magyarázata a 10. ábrán látható. A besatírozott tartományok jelentik $T(\overrightarrow{Q})$-t, ezen belül a sötétebbek $T'(\overrightarrow{Q})$-t. Az e él a $\overrightarrow{Q}$ jobb oldalán van, mert mindkét oldala $T(\overrightarrow{Q})$-beli tartomány. A $\overrightarrow{Q}$ v'-út belseje az s-től t-ig terjedő része. A továbbiakban, ha egy él egyik oldalán lévő tartomány $T(\overrightarrow{Q})$-beli, azt az él mellé tett üres körrel jelöljük az ábrákon. A nem $T(\overrightarrow{Q})$-beli tartományok jele x. Az $\overrightarrow{e,f}$ és $\overrightarrow{f,e}$ él-intervallumok szemléletes jelentése is látható az ábrán. Fontos, hogy e és f nincsenek benne egyik intervallumban sem!

4.21. Lemma. *Legyen $\overrightarrow{Q}$ tetszőleges v'-út $s \in t_0$-ból $t \in t_0$-ba, legyen $\overrightarrow{R}$ v'-út a $p \in P(s,t)$ pontból a $q \in P(t,s) - Q$ pontba. Ekkor $\overrightarrow{R}$ út belseje használ a $\overrightarrow{Q}$ jobb oldalából kimenő élet.*



10. ábra: A $T(Q)$ definíciója és az él-intervallum jelentése.

Bizonyítás. Legyen $T' = T'(\overrightarrow{Q})$ és legyen $T = T(\overrightarrow{Q})$. Egy él T-él, ha mindkét oldalán lévő tartomány T-beli. Könnyen látható, hogy egy v'-re illeszkedő él pontosan akkor T-él, ha a másik végpontja $P(t,s) - \{t,s\}$-beli. A $\overrightarrow{(q,v')}$ él tehát T-él, és mivel $q \notin \overrightarrow{Q}$, ezért a $\overrightarrow{(q,v')}$-t $\overrightarrow{R}$-ben megelőző él is T-él. A $\overrightarrow{(v',p)}$ él nem T-él, így az $\overrightarrow{R}$ út élein végigmenve vehetjük a legutolsó nem T-élet $\overrightarrow{e} = \overrightarrow{(u,v)}$-t, és az utána következő T-élet, $\overrightarrow{f} = \overrightarrow{(v,w)}$-t. A fentiek alapján $\overrightarrow{f} \neq \overrightarrow{(q,v')}$, tehát f az $\overrightarrow{R}$ belsejében van.

Azt állítjuk, hogy $\overrightarrow{f}$ a $\overrightarrow{Q}$ jobb oldalából kimenő él. Ehhez csak azt kell belátnunk, hogy $v \in \overrightarrow{Q}$. Ez azonban triviális, ugyanis ha nem így lenne, akkor v-re nem illeszkedne $\overrightarrow{Q}$-beli él, és így a v-re illeszkedő összes tartomány egyszerre lenne T-beli vagy nem T-beli tartomány. Mivel $\overrightarrow{f}$ mindkét oldala T-beli, így az összes v-re illeszkedő tartomány T-beli lenne, és így $\overrightarrow{e}$ két oldala is T-beli lenne, ellentmondásban $\overrightarrow{e}$ választásával. $\square$

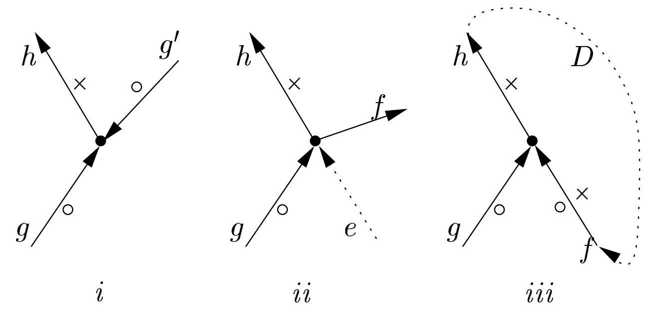
4.22. Lemma. *Ha $\overrightarrow{Q} = \{\overrightarrow{e_1} = \overrightarrow{(v',s)}, \overrightarrow{e_2}, \ldots, \overrightarrow{e_l} = \overrightarrow{(t,v')}\}$ leg-bal vissza út, és belseje nem tartalmaz jobb-kört, akkor minden élének jobb oldala $T(\overrightarrow{Q})$-beli.*

Bizonyítás. A $T'(\overrightarrow{Q})$ halmaz definíciója szerint $\overrightarrow{e_1}$ jobb oldala $T = T(\overrightarrow{Q})$-beli. Tegyük fel, hogy az állítás nem teljesül, legyen $\overrightarrow{h} = \overrightarrow{(v,w)}$ az első él, amelynek jobb oldala nem T-beli ($\overrightarrow{h} \neq \overrightarrow{e_1}$), legyen $\overrightarrow{g} = \overrightarrow{(u,v)}$ a $\overrightarrow{h}$-t megelőző él $\overrightarrow{Q}$-ban. Azt mondjuk, hogy e *korábbi* az f-nél, ha $\overrightarrow{Q}$-n végigmenve hamarabb következik. Egy $\overrightarrow{f}$ él *bemenő él*, ha végpontja v.

A $\overrightarrow{h,g}$ él-intervallum nem lehet üres, hiszen g jobb oldala T-beli, h jobb oldala pedig nem T-beli. Legyen $\overrightarrow{f}$ a $\overrightarrow{h,g}$ jobb-sorrendben utolsó $\overrightarrow{Q}$-beli tagja, ekkor $\overrightarrow{f,g}$-ben nincs $\overrightarrow{Q}$-beli él.

4.23. Állítás. *A* $\overrightarrow{h,g}$*-ben nem lehet* g*-nél korábbi bemenő él.*

Bizonyítás. Ha mégis lenne ilyen g' él, akkor a leg-bal vissza út megkonstruálása során h után nem g-t kellett volna bevenni, hanem g'-t, hiszen az „balrább” van g-től h-hoz képest (11. ábra *i*. része.) $\square$



11. ábra: A 4.23.-4.25. állítások bizonyítása.

4.24. Állítás. *Az* f *él nem lehet* h*-nál későbbi kimenő él.*

Bizonyítás. Tegyük fel, hogy f mégis h-nál későbbi kimenő él. A leg-bal vissza út építésekor f kiválasztása után nem g-t választottuk, azaz van $\overrightarrow{e}$ bemenő $\overrightarrow{Q}$-beli él az $\overrightarrow{f,g}$ él-intervallumban. Ez ellentmondás, hiszen $\overrightarrow{f,g}$-ben egyáltalán nincs $\overrightarrow{Q}$-beli él (11. ábra *ii*. része.) $\square$

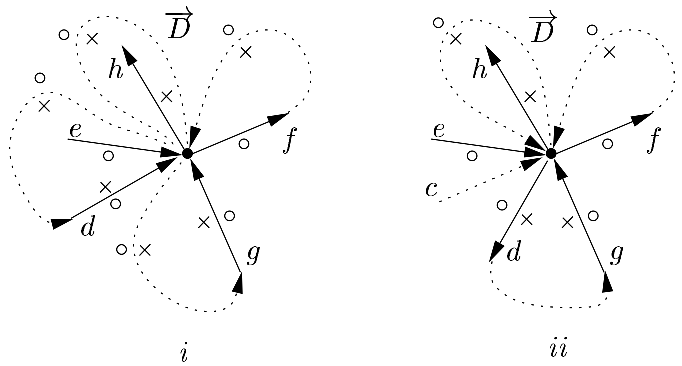
4.25. Állítás. *Az* f *él nem lehet* h*-nál későbbi bemenő él.*

Bizonyítás. Ha mégis így lenne, akkor a $\overrightarrow{Q}$ irányított körséta h-val kezdődő és f-el végződő szakasza egy irányított körséta lenne. Ebből a körből kaphatunk egy egyszerű $\overrightarrow{D}$ kört, amelynek éle f. A g él jobb oldala T-beli, így f bal oldala is T-beli. Ez azt jelenti, hogy az f él bal oldala elérhető t_0-ból a $\overrightarrow{D}$ körre nézve a G gráfban, azaz a 4.17. állítás szerint $\overrightarrow{D}$ jobb-kör, ellentétben a lemma feltételeivel (11. ábra *iii*. része.) $\square$

A fenti három állítás miatt f h-nál korábbi kimenő él. Legyen az f-et $\overrightarrow{Q}$-ban megelőző él e. Ekkor e bemenő él, s így a 4.23. állítás szerint nem lehet $\overrightarrow{h,g}$-ban, tehát $e \in \overrightarrow{g,h}$. Ekkor a $\overrightarrow{Q}$ körséta f-el kezdődő és g-vel végződő része is egy $\overrightarrow{D}$ körséta. Mivel az f él g-nél korábbi, ezért $\overrightarrow{D}$ minden élének jobb oldala T-beli. Legyen d a $\overrightarrow{D}$ $\overrightarrow{f,e}$-be eső jobb-sorrendben legutolsó éle. (Ilyen van, mert $g \in \overrightarrow{f,e}$.) Ekkor $\overrightarrow{d,e}$-ben nincs D-beli él.

4.26. Állítás. *A* d *él nem lehet bemenő.*

Bizonyítás. A d él jobb oldala T-beli, azaz G-ben is elérhető D-re nézve, így bal oldala nem érhető el. Ez ellentmond annak, hogy e jobb oldala T-beli és így elérhető D-re nézve (12. ábra *i*. része.) $\square$



12. ábra: A 4.26. és 4.27. állítások bizonyítása.

4.27. Állítás. *A d él nem lehet kimenő.*

Bizonyítás. A d él e-nél későbbi, és a leg-bal vissza út építésekor d beválasztása után nem e-t választottuk. Ez csak úgy lehet, hogy ha a d-t $\overrightarrow{Q}$-ban – és így $\overrightarrow{D}$-ben – megelőző c él a $\overrightarrow{d,e}$ él-intervallumban van. Ez nem lehetséges, mivel $\overrightarrow{d,e}$-ben nincs $\overrightarrow{D}$-beli él. □

A fenti két állítás szerint d nem lehet kimenő, sem pedig bemenő él, ami ellentmondás. Ezzel a lemmát bebizonyítottuk. □

4.28. Állítás. *Ha $\overrightarrow{R}$ leg-bal vissza út, akkor nincs a jobb oldalába bemenő él.*

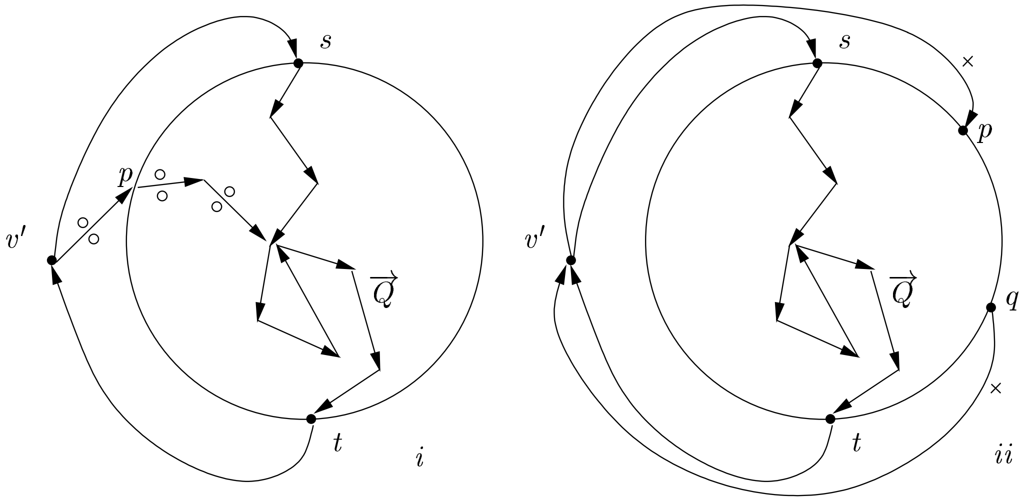
Bizonyítás. Tegyük fel indirekten, hogy $\overrightarrow{e} = \overrightarrow{(u,v)}$ a $\overrightarrow{R}$ jobb oldalába bemenő él, ekkor e nem lehet a $\overrightarrow{Q}$ éle. Legyen f és g az e-től legjobbra illetve legbalra lévő él az $\overrightarrow{R}$ élhalmazára nézve. A f nem lehet v-ből kimenő él, hiszen ekkor a leg-bal vissza út építésekor f után e-t kellett volna bevennünk, de nem lehet v-be bemenő él sem, mert a 4.22. lemma miatt a jobb oldala $T(\overrightarrow{Q})$-beli, így a bal oldala nem $T(\overrightarrow{Q})$-beli, ellentmondásban azzal, hogy e jobb oldala $T(\overrightarrow{Q})$-beli. Az f él tehát nem lehet sem kimenő, sem bemenő él, ami ellentmondás. □

4.29. Lemma. *Legyen Q leg-bal vissza út $s \in t_0$-ből $t \in t_0$-ba, aminek belseje nem tartalmaz jobb-kört. Legyen R olyan v'-út ami a $P(s,t) \cup (P(t,s) - - Q)$ két pontját köti össze, és amelynek belseje használ $\overrightarrow{Q}$ jobboldalából kimenő élet. Ekkor az $\overrightarrow{R}$ út a $P(s,t)$ halmazból a $P(t,s) - Q$ halmazba megy.*

Bizonyítás. Legyen $T = T(\overrightarrow{Q})$, és nevezzünk egy élet T-élnek, ha mindkét oldala T-beli.

Tegyük fel indirekten, hogy az $\overrightarrow{R}$ út a $p \in P(t,s) - Q$ pontból indul, és legyen $\overrightarrow{e}$ az út legelső olyan éle, amelynek végpontja a $\overrightarrow{Q}$ úton van. Az $\overrightarrow{R}$

út e-ig tartó szakasza T-élekből áll, így e a $\overrightarrow{Q}$ jobb oldalába bemenő él, ami a 4.28. állítás szerint lehetetlen (13. ábra i. része.)



13. ábra: Az R út $P(s,t)$-ből $P(t,s) - Q$-ba megy.

$\overrightarrow{R}$ tehát $p \in P(s,t)$-ből indul. Tegyük fel, hogy az állítással ellentétben $q \in P(s,t)$-be érkezik. Ekkor sem $\overrightarrow{(v',p)}$, sem $\overrightarrow{(q,v')}$ nem T-él. Az $\overrightarrow{R}$ úton viszont van T-él, hiszen van $\overrightarrow{Q}$ jobb oldalából kimenő él. Legyen az $\overrightarrow{R}$ út utolsó T-éle $\overrightarrow{e}$. Ekkor $\overrightarrow{e}$ végpontja $\overrightarrow{Q}$-n kell, hogy legyen, különben az e után következő él is T-él lenne. Ez azt jelenti, hogy $\overrightarrow{e}$ a $\overrightarrow{Q}$ jobb oldalába bemenő él, ellentétben a 4.28. állítással (13. ábra ii. része.) Ezzel a lemmát bebizonyítottuk. □

4.30. Lemma. *Tegyük fel, hogy a gráfban nincs jobb-kör. Legyen $\overrightarrow{P}$ v'-út $p \in t_0$-ból $q \in t_0$-ba, és legyen $\overrightarrow{Q}$ leg-bal vissza út $p' \in t_0$-ból q-ba úgy, hogy p' nincs rajta $\overrightarrow{Q}$-n. Ekkor p' a $P(q,p)$ íven van.*

Bizonyítás. A lemma a 4.29. lemma triviális alkalmazása $R = P$ választással. □

4.3.3. Az algoritmus

4.31. Tétel. *Legyen $G = (V,E)$ irányított síkgráf, $H = (V,I)$ igénygráf, amelyben $I = \{\overrightarrow{(t_i,s_i)} : i = 1,...,k\}$. Tegyük fel, hogy $G + H = (V, E \cup I)$ Euler, és G-ben nincs jobb-kör. Legyen $C = \{v_0, v_1, \ldots, v_l\}$ a végtelen lapot határoló kör, úgy, hogy az éleit $\overrightarrow{(v_i,v_{i+1})}$ irányban megirányítva jobb-kört kapjunk. Tegyük fel továbbá, hogy minden i igényre $s_i = v_{a_i} \in C$ és $t_i = = v_{b_1} \in C$, azaz az igényélek a külső ablak pontjait kötik össze, továbbá $b_i < a_i$ és $b_i < b_j$ ha $i > j$.*

Ekkor az élidegen út probléma pontosan akkor megoldható, ha a vágásfeltétel teljesül. Az útrendszer, illetve egy vágásfeltételt sértő halmaz polinomiális időben megtalálható.

Bizonyítás. Adunk egy algoritmust, amely megad egy megfelelő útrendszert ha létezik, vagy előállít egy, a vágásfeltételt sértő halmazt. Az algoritmus mohó módon egymás után keres utat az egyes igényeknek.

Vegyünk fel a C által határolt külső t_0 tartományban egy v' pontot és húzzuk be minden i. igényre a $\overrightarrow{(v', s_i)}$ és $\overrightarrow{(t_i, v')}$ éleket. Az így kapott $G' = (V', E')$ gráf a tétel feltételei szerint Euler.

Az algoritmus fázisokból áll, az i. fázisban az i. igénynek keresünk utat, az előző fázisokban megtalált utakat jelölje $P_1, \ldots, P_{i-1}$. Az útkeresés a következőképpen történik: csak azt az élhalmazt tekintjük, amit az előzőleg talált utak nem használnak: $E'' = E' - \bigcup_{j=1}^{i-1} P_j$. Ekkor legyen P_i a $\overrightarrow{(v', s_i)}$ éllel kezdődő leg-jobb út E''-ben. Az útépítés nem akadhat el, így visszaérünk v'-be, azaz egy v'-utat kapunk, amelynek utolsó éle $\overrightarrow{(t_j, v')}$ valamely j-re.

Jelölje P_i' azt az utat, amit a P_i körséta első és utolsó – v'-re illeszkedő – éleinek elhagyásával kapunk. Amennyiben minden $i - re$ a P_i' út utolsó pontja t_i, akkor megtaláltuk a keresett éldiszjunkt utakat.

Ellenkező esetben legyen i a legkisebb index, amelyre a P_i' nem t_i-be érkezik. Ekkor $j = 1, ..., i-1$-re a P_j' út t_j-be megy. A P_i' út s_i-ből indul, és $t_{i+\delta}$-ba érkezik valamely $\delta > 0$-ra $(t_{i+\delta} \neq t_i)$.

Tekintsük most az első i út által meghatározott E^* élhalmazt. Legyen $\overrightarrow{Q} = \{\overrightarrow{e_1}, \overrightarrow{e_2}, \ldots, \overrightarrow{e_m}\}$ a $\overrightarrow{(t_{i+\delta}, v')}$ éllel végződő leg-bal vissza út E^*-ban. A $\overrightarrow{Q}$ első éle ekkor $\overrightarrow{(v', s_j)}$ valamely $j \leq i$-re. A 4.21. állítás szerint minden $S' := P(s_j, t_{i+\delta})$-ból $T' := P(t_{i+\delta}, s_j) - Q$-ba menő v'-út használ élet a $\overrightarrow{Q}$ jobb-oldalából.

4.32. Állítás. *$s_i \in S'$ és $t_i \in T'$.*

Bizonyítás. Az állítás triviális, ugyanis a 4.30. állítást ha a $\overrightarrow{Q}$-ra és P_i-re alkalmazzuk, azt kapjuk, hogy $s_i \in P(s_j, t_{i+\delta}) = S'$. Legyen $v_{i'} = s_i, \quad v_{i''} = t_i, \quad v_{j'} = s_j, \quad v_{j''} = t_j, \quad v_{\delta'} = t_{i+\delta}$ a C kör pontjai a tétel kimondása szerint. Ekkor a feltételek szerint $\delta' < i'' \leq j'' < j'$, vagyis $t_i \in P(t_{i+\delta}, s_j)$, ahogy akartuk. $\square$

4.33. Állítás. *Ha $\overrightarrow{e}$ a $\overrightarrow{Q}$ jobb oldalából kimenő él, akkor a $P_1', \ldots, P_{i-1}'$ utak valamelyike használja $\overrightarrow{e}$-t.*

Bizonyítás. Tegyük fel indirekten, hogy $\overrightarrow{e} = \overrightarrow{(u, v)}$-t nem használja az első i út.

Járjuk körül az u-ra illeszkedő éleket jobb-sorrendben e-vel indulva, és legyen f és g az első, illetve az utolsó $\overrightarrow{Q}$-beli él. A 4.22. lemma szerint f és g jobb oldala $T(\overrightarrow{Q})$-beli, így $\overrightarrow{f}$ u-ba bemenő, míg $\overrightarrow{g}$ u-ból kimenő él kell, hogy legyen. Mivel a $\overrightarrow{Q}$ út keresését az $E^* = \bigcup_{i=1}^{k} P_i$ élhalmazon végeztük, így g

is éle valamelyik útnak, például P_l-nek. Legyen h a P_l-ben g-t megelőző él. A h él nem lehet $\overrightarrow{e,g}$-ben, mert ekkor h után e-t kellett volna választanunk g helyett, hiszen P_l leg-jobb út. Ugyanakkor nem lehet $\overrightarrow{g,e}$-ben sem, különben g után nem f-et kellett volna a $\overrightarrow{Q}$ leg-bal útba beválasztani, hanem h-t (ha $h \in \overrightarrow{g,e}$, akkor nem lehet $\overrightarrow{Q}$-beli f és g választása miatt.) Ezzel az indirekt feltevést megcáfoltuk, már csak azt kell megmutatnunk, hogy nem P_i használja e-t, ez azonban a 4.29. és 4.32. állítások miatt világos. □

Az előző és a 4.29. állítások összevetésével adódik, hogy minden $\overrightarrow{Q}$ jobb oldalából kimenő élhez létezik egy s_l, t_l $(1 \leq l < i-1)$ igény S'-ből T'-be, amely használja az élet.

Jelölje k' az $(s_1, t_1), \ldots, (s_{i-1}, t_{i-1})$ igények közül azok számát, amelyek S'-ből T'-be mennek, és legyen Q_{ki} a $\overrightarrow{Q}$ jobb oldalából kimenő élek halmaza. Azt láttuk tehát, hogy $|Q_{\mathrm{ki}}| = k'$. Ez azonban azt jelenti, hogy a Q_{ki} élhalmaz sérti a vágásfeltételt, mivel megmutattuk, hogy $|Q_{\mathrm{ki}}| + 1$ igénynek kell használni Q_{ki} éleit, hiszen a 4.32. állítás miatt az i. igény is S'-ből T'-be megy. Ezzel a tétel bizonyítását befejeztük. □

A 1. ábra a 2.2. fejezetben azt bizonyította, hogy a vágásfeltétel nem elég a törtfolyam létezéséhez. Ugyanez a gráf azt is bizonyítja, hogy a jobb-körök kizárása szükséges, hiszen az ábrán látható gráf a tétel minden feltételét teljesíti, kivéve azt, hogy nincs benne jobb-kör.

4.3.4. Az irányítatlan változat

A 4.31. tétel alkalmazásaként egyszerű bizonyítást nyerhetünk az Okamura-Seymour tételre, amely a feladat irányítatlan változatára vonatkozik. A tétel teljesen más megközelítést használó eredeti bizonyítása [7]-ben található.

4.34. Tétel. *Legyen G irányítatlan síkbarajzolható gráf, H igénygráf, amiben az igényélek a külső ablak pontjait kötik össze. Tegyük fel, hogy $G+H$ Euler, ekkor pontosan akkor léteznek az igényeket kielégítő éldiszjunkt utak, ha teljesül a vágásfeltétel.*

Bizonyítás. A vágásfeltétel szükségessége nyilvánvaló. Tegyük most fel, hogy teljesül a vágásfeltétel. Készítünk egy irányított gráfot G-ből, amely teljesíti a 4.31. tétel feltételeit. Tekintsük tehát a G gráf síkbarajzolását, amelyben a külső ablak $C = \{v_0, \ldots, v_{l-1}\}$ olyan, hogy $\overrightarrow{(v_i, v_{i+1})}$ irányítással jobb-kört kapunk. Számozzuk át és irányítsuk meg az igényéleket úgy, hogy a 4.31. tétel s_i, t_i párokra vonatkozó sorrendiségi feltételei teljesüljenek.

Most megirányítjuk a G gráfot úgy, hogy irányított értelemben is Euler legyen. Legyen a v pont foka $G+H$-ban $d_{G+H}(v)$, legyen az igények fenti irányítása szerint befoka és kifoka $\varrho_H(v)$ és $\delta_H(v)$. Legyen $m(v) := \frac{d_{G+H}(v)}{2} - - \varrho_H(v)$. Jelölje $i_G(X)$ a G gráfban az X ponthalmaz által feszített élek számát, $d_G(S,T)$ az S és T ponthalmazok között menő élek számát, és legyen $m(X) := \sum_{v\in X} m(x)$. Közismert az alábbi lemma:

4.35. Lemma. *Legyen $m : V \to Z^+$ függvény. Pontosan akkor létezik $G = (V,E)$-nek olyan irányítása, amelyben minden $v \in V$-re $\varrho(v) = m(v)$, ha $m(V) = |E|$ és minden $X \subset V$-re $i_G(X) \le m(X)$.* □

A mi esetünkben $m(X) = \sum_{v\in V} \frac{d_G(v)}{2} + \sum_{v\in V}(\frac{d_H(v)}{2} - \varrho_H(v)) = |E|$. Továbbá minden $X \subset V$-re

$$m(X) = \sum_{v\in X} \frac{d_G(v)}{2} + \sum_{v\in X} \left(\frac{d_H(v)}{2} - \varrho_H(v)\right) \ge$$
$$\ge i_G(X) + \frac{1}{2} d_G(X, V-X) - \frac{1}{2} \sum_{\substack{u\notin X, v\in X\\(u,v)\in H}} 1$$

A vágásfeltétel miatt az X és $V-X$ között menő igényélek száma legfeljebb $d_G(X, V-X)$, így a lemma feltételei teljesülnek.

Van tehát egy irányításunk, amellyel G Euler. Ebből az irányításból nyerünk egy olyan irányítást, amely szerint nincsenek jobb-körök.

4.36. Állítás. *Legyen $G = (V,E)$ irányított síkgráf. G megirányítható úgy, hogy az új irányításban nincsenek jobb-körök, és a pontok befoka megegyezik az eredeti befokukkal.*

Bizonyítás. Ha van a gráfban jobb-kör, az azt jelenti, hogy nem minden tartomány érhető el a duális gráfban a t_0 tartományból irányított úton.

Tekintsük tehát a duálisban t_0-ból elérhető tartományok S halmazát. Ha S az összes tartományt tartalmazza, akkor készen vagyunk, a gráfban nincs jobb-kör. Egyébként S olyan irányított vágást határoz meg, amelyből nem indul ki él, azaz az S-nek megfelelő kör az eredeti gráfban jobb-kör. Fordítsuk meg ennek a körnek az irányítását, ekkor a ki- és befokok változatlanok maradnak, azonban t_0-ból több tartomány lesz elérhető irányított úton. Folytassuk az eljárást, amíg minden tartomány elérhető nem lesz, azaz a gráfban már nem lesz jobb-kör. □

Futtassuk le az így kapott irányított gráfra a 4.31. tételben szereplő algoritmust. Ha az algoritmus talál megoldást, az az irányítatlan feladatnak is megoldása. Ha az algoritmus egy sértő vágást ad, akkor ez a vágás az irányítatlan problémában is sérő vágás, hiszen a megkonstruált vágás irányított. Ezzel a tétel bizonyítását befejeztük. □

A tétel egyik fontos következménye a következő:

4.37. Következmény. *Legyen G irányítatlan síkgráf, legyen H igénygráf olyan, hogy élei a G külső ablakának pontjait kötik össze. Ekkor pontosan akkor létezik az igényeket kielégítő törtfolyam, ha teljesül a vágásfeltétel. Ha létezik folyam, akkor létezik fél-egész folyam is.*

Bizonyítás. Duplázzuk meg a gráf éleit, és az igényéleket. Az így keletkező $G' + H'$ gráf Euler, így teljesíti az 4.34. tétel feltételeit. A vágásfeltétel szükségessége nyilvánvaló. Ha G-ben teljesül a vágásfeltétel, akkor G'-ben is, ott léteznek éldiszjunkt utak, minden eredeti igényre két éldiszjunkt utat kapunk, a folyamot leosztva G-ben egy fél-egész folyamot kapunk. □

4.4. Approximációs algoritmus irányítatlan gyűrűre

Ebben a fejezetben a Minimális terhelés irányított gyűrűn feladat irányítatlan változatára adunk approximációs algoritmust Schrijver, Seymour és Winkler [8] cikke alapján. Az irányított változathoz képest lényeges különbség az, hogy itt a gyűrű és az igények irányítatlanok, illetve nem csak 1 nagyságú igényeket engedünk meg, hanem tetszőlegeseket.

4.38. Definíció. *Egy n pontú irányítatlan gyűrűn a következő gráfot értjük: $G = (V, E) = (\{0, 1, \ldots, n-1\}, \{(0,1), (1,2), \ldots, (n-1, 0)\})$. Az előre és hátra ívek definíciója ugyanaz, mint az irányított esetben.*

Minimális terhelés irányítatlan gyűrűn
Adott: Egy n pontú irányítatlan gyűrű és k darab igény s_i, t_i, d_i $(i = 1, ..., k)$ (egy pontpárra csak egy igény adott, minden igényre $s_i < t_i$.)
Feladat: Adjuk meg azt a minimális L számot, hogy ha minden él kapacitása L, akkor létezik osztatlan folyam, amely kielégíti az igényeket.

4.4.1. Az algoritmus és helyessége

Az irányított esethez hasonlóan itt is először a lineáris programozási relaxációt oldjuk meg. Jelölje tehát $x_i^* \in [0, d_i]$, hogy az i. igényből mennyi megy az előre úton, ekkor az $(l, l+1)$ él terhelése (a pontokkal végzett műveleteket modulo n tekintjük)

$$L_l^* = \sum_{l \in [s_i, t_i - 1]} x_i^* + \sum_{l \in [t_i, s_i - 1]} d_i - x_i^* \tag{25}$$

A lineáris program célja pedig, hogy $L^* = \min L_l^*$-ot minimalizálja.

A törtmegoldás megtalálásához azonban nem kell megoldani a lineáris programot, ugyanis az Okamura-Seymour tétel 4.37. következményének segítségével közvetlen feltételt adhatunk a törtfolyam létezésére. A gráf síkgráf, az igények is a külső ablak pontjait kötik össze, így 4.37 feltételei teljesülnek. Legyen tehát minden él kapacitása L, ekkor pontosan akkor létezik törtfolyam, ha teljesül a vágásfeltétel.

Jelölje $D_{g,h}$ azon igények összegét, amelyek a $(g, g+1), (h, h+1)$ vágás két oldala között mennek:

$$D_{g,h} = \sum (d_i : s_i \text{ és } t_i \text{ közül pontosan egy esik } [g+1, h]\text{-ba.})$$

Ekkor a vágásfeltétel alakja: $2L \geq \max_{g,h} D_{g,h}$. Ebből következően a tört optimum $L^* = \frac{1}{2}\max D_{g,h}$ amely mennyiség $O(n^2)$ időben meghatározható. Ezután az 4.37-ben leírt algoritmussal meghatározunk egy fél-egész folyamot.

Következő lépésben a irányított esethez hasonlóan kikeresztezzük a párhuzamos igényeket.

4.39. Definíció. *Az s, t, d és s', t', d' igények keresztezőek, ha az s és t közül pontosan az egyik van az (s', t') nyílt intervallumban. Különben párhuzamosak. A $(g, g+1)$ él a két párhuzamos igény között van, ha nem lehet benne olyan vágásban, amely mindkét igényt elválasztja.*

Legyenek s_i, t_i, d_i és s_j, t_j, d_j osztott párhuzamos igények, $(g, g+1)$ köztük lévő él. Ekkor ha $(g, g+1)$-en az első igényből x_i, a másodikból $d_j - x_j$ megy, akkor ezek minimumával növelve a nagyobbat és csökkentve a kisebbet az irányított esethez hasonlóan elérhető, hogy az élterhelés sehol sem nőjön, de eggyel kevesebb legyen az osztott igény. Legfeljebb k kikeresztezési lépés után már nem lesznek osztott párhuzamos igények.

Jelölje most x_i^* a kikeresztezés után kapott irányítást. Feladatunk a továbbiakban az x^*-ot egy olyan x vektorrá alakítani, hogy minden i-re $x_i \in \{0, d_i\}$, azaz amely irányítás egy igényt teljes egészében előre, vagy hátra küld.

Feledkezzünk meg most a nem osztott igényekről, és az általuk okozott élterhelésről, ugyanis az algoritmus további lépéseiben csak az osztott igényekhez tartozó változókat változtatjuk. A továbbiakban tehát L^* az osztott igények által okozott élterhelés. Mivel nincsenek párhuzamos osztott igények, ezért legfeljebb n darab osztott igény lehet.

Amennyiben a gyűrű egy i pontja nem végpontja osztott igénynek, akkor azt elhagyhatjuk, a két szomszédját $i-1$-et és $i+1$-et pedig összeköthetjük egy éllel. Ezen az élen legyen a terhelés értéke $\max\{L_{i-1}^*, L_i^*\}$. Ha az így kapott gráfon irányítjuk meg az osztott igényeket, akkor az így kapott terhelés érték legalább akkora lesz, mintha az eredeti gráfban tekintenénk az irányítást. Miután minden nem használt pontot így kitöröltünk, a gráfnak $2m$ pontja lesz, ahol m az osztott igények száma, az i. igény az $i, i+m$ pontok között fog menni. Legyen $D = \max_i d_i$ a maximális osztott igény. Schrijver, Seymour és Winkler [8]-ban bizonyították a következőt:

4.40. Tétel. *Adott egy $2m$ pontú irányítatlan gyűrű, m darab igény: $i, i+$ $+m, d_i$ alakúak, és egy x^* irányítás L^* terheléssel. Ekkor polinom időben adható olyan osztatlan x irányítás, amelynek L élterhelésére $L \leq L^* + \frac{3}{2}D$.*

Bizonyítás. Legyen x egy osztatlan irányítás, ekkor az $(l, l+1)$ él terhelése a következőképpen változik:

$$L_l = L_l^* + \sum_{\substack{i\in[1,m]\\ l\in[i,i+m-1]}} (x_i - x_i^*) - \sum_{\substack{i\in[1,m]\\ l\in[i+m,i-1]}} (x_i - x_i^*)$$

Ha $l \in [1, m-1]$, akkor az első szummában az $1 \ldots l$ igények, a másodikban az $l+1, ..., m$ igények szerepelnek. Ha $l \geq m$, akkor az első szummában az $l-m+1 \ldots m$ igények vannak benne, a másodikban pedig a $1, ..., l$ igények. Elmondható tehát, hogy

$$\begin{aligned} |L_l - L_l^*| \leq \max_j \left| \sum_{i=1}^{j} (x_i - x_i^*) - \sum_{i=j+1}^{m} (x_i - x_i^*) \right| = \\ \left| 2\sum_{i=1}^{j} (x_i - x_i^*) - \sum_{i=1}^{m} (x_i - x_i^*) \right| \end{aligned} \tag{26}$$

Válasszuk most x_j-t rekurzívan a következőképpen:

$$x_j = \begin{cases} d_j & \text{ha } -x_j^* + \sum_{i=1}^{j-1}(x_i - x_I^*) < -\frac{D}{2} \\ 0 & \text{különben} \end{cases}$$

Ilyen választás mellett minden j-re teljesül, hogy $\left|\sum_{i=1}^{j}(x_i - x_i^*)\right| \leq \frac{D}{2}$. Ezzel megbecsülve (26) jobboldalát, kapjuk, hogy $|L_j - L_j^*| \leq 3\frac{D}{2}$, vagyis a kapott irányítás legfeljebb $\frac{3}{2}D$-vel nagyobb terhelést ad az optimumnál. Ezt kellett igazolnunk. □

Az eredeti problémát tekintve ez azt jelenti, hogy az osztott igények osztatlanná tétele a maximális élterhelést legfeljebb $\frac{3}{2}D$-vel növeli meg, így a következőt bizonyítottuk be:

4.41. Tétel. *A* Minimális terhelés irányítatlan gyűrűn *problémára létezik polinomiális algoritmus, amely olyan irányítást ad, amelynek terhelése legfeljebb $\frac{3}{2}D$-vel nagyobb az optimálisnál. [8]* □

Érdemes kimondani a tételt egy másik formában is, amely jobban tükrözi a bizonyítás gondolatát:

4.42. Tétel. *Ha adott a probléma egy törtmegoldása, akkor polinomiális időben megtalálható egy olyan osztatlan megoldás, amelynek terhelése legfeljebb $\frac{3}{2}D$-vel nagyobb a törtmegoldás terhelésénél.* □

4.4.2. A probléma ϵ-közelítése

Az előző fejezetben ismertetett algoritmus felhasználásával egyszerű ϵ-közelítő algoritmus sémát adhatunk a minimális terhelés irányítatlan gyűrűn feladatra. A fejezetben ismertetendő eredmény Sanjeev Khanna munkája [10].

4.43. Definíció. *A gyűrű két pontját összekötő hosszabb úton a két lehetséges közül a hosszabbat értjük, ha ez nem egyértelmű, akkor az előre utat.*

Az ϵ-közelítés alapja a következő egyszerű észrevétel:

4.44. Állítás. *Legyen Γ a hosszabb úton elszállított igények összege egy tetszőleges osztatlan megoldásban. Ekkor van él, amelynek terheltsége $\Gamma/2$.*

Bizonyítás. Az állítás nyilvánvaló, hiszen ekkor az összes élkihasználtság legalább $\Gamma\lceil\frac{n}{2}\rceil$, így kell, hogy legyen él, amely terhelése legalább $\Gamma/2$. □

Jelölje az optimális megoldást L_{OPT}. Legyen az igények halmaza S. Az igényeket két részre osztjuk a következő módon: legyen S_{nagy} a legalább $\frac{2}{3}\epsilon L_{OPT}$ nagyságú igények halmaza, a kis igényeket, azaz a maradékot jelölje S_{kicsi}. A fenti állításba behelyettesítve kapjuk, hogy:

4.45. Következmény. *Egy optimális megoldásban legfeljebb $3/\epsilon$ darab S_{nagy}-beli igény mehet a hosszú úton.* □

Tegyük fel most, hogy ismerjük L_{OPT} értékét. Az algoritmus alapgondolata, hogy végigpróbálgatjuk az S_{nagy} összes legfeljebb $3/\epsilon$ méretű részhalmazát, és a maradék nagy igényeket illetve a kicsiket ettől függően irányítjuk meg. Amennyiben ϵ-t rögzítjük, akkor ez n-ben polinomiális sok részhalmazt jelent (ϵ-nal 0-hoz tartva a futási idő persze exponenciálisan nő.) Az algoritmus tehát:

- `Az` S_{nagy} `összes legfeljebb` $3/\epsilon$ `elemű` S' `részhalmazára:`
    - `Irányítsuk az` S'`-beli igényeket a hosszabb úton, az` $S_{\text{nagy}} - S'$`-belieket a rövidebb úton. Jelölje ezek után az` $(i, i+1)$ `él terhelését` L_i`.`
    - `Keressük meg azt a minimális` c `számot, hogy a` $c - L_i$ `kapacitásfüggvény mellett létezzen törtfolyam, amely kielégíti az` S_{kicsi}`-beli igényeket. (Ezt a` c`-t lineáris programozással, vagy a 4.37. tétel használatával kombinatorikus módon is megkereshetjük.)`
    - `A törtmegoldásból kiindulva használjuk a 4.42. tételbeli algoritmust, hogy az` S_{kicsi}`-beli igényeknek egy osztatlan elvezetését kapjuk.`

A végigpróbált esetek közül természetesen a legkisebb terhelésű lesz az algoritmus eredménye. Mivel az optimális megoldás legfeljebb $3/\epsilon$ darab nagy igényt visz a hosszú úton, így az algoritmus egyik lépésében pontosan azokat a nagy igényeket irányítjuk a hosszabb úton, mint amit az optimális megoldás. Az ezután megtalált c szám legfeljebb annyi, mint L_{OPT} és így a 4.42. tétel közelítése alapján az eredményül adódó terhelés legfeljebb $\frac{3}{2}(\frac{2}{3}\epsilon L_{OPT})$-al nagyobb az optimálisnál, azaz az algoritmus egy $(1+\epsilon)$-közelítést állít elő.

A szóbajövő részhalmazok M számára:

$$M = \sum_{i=0}^{3/\epsilon} \binom{|S_{\text{nagy}}|}{i} \leq \frac{3}{\epsilon} \left(\frac{e|S_{\text{nagy}}|}{3/\epsilon} \right)^{3/\epsilon}$$

amit továbbecsülve a triviális $|S_{\text{nagy}}| \leq \binom{n}{2}$ egyenlőtlenséggel és feltéve, hogy $\epsilon < 1/2$ kapjuk, hogy $M \leq \frac{3}{\epsilon} n^{6/\epsilon}$. Belátható, hogy az egy halmaz vizsgálata során végzett műveletek $O(n^4)$ időben megvalósíthatók, így az algoritmus futási ideje $O(\frac{1}{\epsilon} n^{\frac{6}{\epsilon}+4})$. Az M óvatosabb becslésével $O(\frac{1}{\epsilon} n^{\frac{3}{\epsilon}+4})$ is megmutatható.

Hátra van még annak a feltevésnek az eltüntetése, hogy L_{OPT} értékét ismerjük. Ehhez megbecsüljük L_{OPT}-ot és a becslésünket használjuk. Egyszerű észrevétel, hogy $L^* \leq L_{OPT} \leq 2L^*$, ahol L^* a relaxált optimum. Az algoritmust sorra lefuttatjuk az L_{OPT} különböző becsléseire. Legyen $L_i = \frac{i}{n} L^*$ az i. becslés, ahol $i = n, n+1, \ldots, 2n$. Ekkor lesz egy olyan j pillanat, amikor $L_{OPT} \leq L_j \leq (1+1/n)L_{OPT}$. Ha ezzel az L_j becsléssel futtatjuk az algoritmust, akkor az elemzés során tett megállapításaink továbbra is ugyanúgy működni fognak, azzal az eltéréssel, hogy nem $1+\epsilon$-közelítést, hanem $1+(1+1/n)\epsilon$ közelítést kapunk. Az algoritmust tehát $\epsilon' := \epsilon/(1+1/n)$-el futtatva $(1+\epsilon)$-közelítő algoritmust kapunk, amelynek futási ideje $O(\frac{1}{\epsilon} n^{\frac{3(1+1/n)}{\epsilon}+5})$.

Az alábbi tételt bizonyítottuk be tehát (a futási időre vonatkozóan gyengébbet bizonyítottunk):

4.46. Tétel. *Létezik a* Minimális terhelés irányítatlan gyűrűn *feladatra ϵ-közelítő algoritmus séma, amelynek futási ideje $O(\frac{1}{\epsilon} n^{\frac{3(1+1/n)}{\epsilon}+5})$. [10]*

4.5. Approximációs algoritmus az egyforrásos problémára

A gyűrű topológia után egy általánosabb eredmény ismertetünk, amely tetszőleges irányított gráfban igaz, azonban erős feltevéssel él az igények elhelyezkedéséről. A feladat a következő:

EGYFORRÁSOS OSZTATLAN FOLYAM PROBLÉMA
Adott: $G = (V, E)$ irányított gráf, c kapacitásfüggvény, $s \in V$ kijelölt pont, és k darab igény, amelyeknek a forrása közös: (s, t_i, d_i), $i = 1, ..., k$.
Feladat: Adjunk osztatlan folyamot, amely az igényeket kielégíti.

Ez a feladat – bár speciálisabb az osztatlan folyam problémánál – szintén NP-nehéz, hiszen a 4.1. tételben a hátizsákpakolási feladatból konstruált osztatlan folyam probléma egyforrásos volt.

4.5.1. Az algoritmus

Ebben a fejezetben célunk Dinitz, Garg és Goemans [9] algoritmusának bemutatása, amely egy törtfolyamból indul ki és osztatlanná teszi úgy, hogy közben az élek kapacitása legfeljebb a maximális igénnyel sérül. Jelölje $c_{\min}$ a minimális élkapacitást, $d_{\min}$ a minimális (nem nulla) igényt, $d_{\max}$ pedig a maximális igényt.

Az algoritmus egy – az összes igényt kielégítő – f megengedett törtfolyamból indul ki. Ilyen folyam Ford és Fulkerson eredeti maximális folyam algoritmusával is található. Feltehető, hogy a folyam aciklikus, azaz a nem 0 értékű ($f(e) \neq 0$) éleket tekintve nincs benne irányított kör. Az algoritmus ezt a folyamot fogja módosítani, mozgatni fogja a nyelőket és törli a 0 értékű éleket. Ezen a ponton elfeldkezhetünk az élek eredeti kapacitásáról, a továbbiakban csak a folyammal fogunk dolgozni.

Az eljárás során végig fenntartjuk azt az invariáns tulajdonságot, hogy az aktuális (nem feltétlenül osztatlan) folyam kielégíti az aktuális igényeket.

- **Első lépés.** Töröljük azon éleket, amin a folyam értéke 0.
- **Második lépés.** Minden t_i nyelőre ellenőrizzük, hogy létezik-e $e = (v, t_i)$ él, amelyre $f(e) \geq d_i$. Ha létezik ilyen, akkor a t_i nyelőt a v pontba mozgatjuk, az e élen pedig csökkentjük a folyamértéket d_i-vel. Ha az él értéke 0 lesz, akkor töröljük a gráfból. Ha az így módosított gráfban van osztatlan folyam, akkor ez könnyen kiterjeszthető az eredeti gráf egy osztatlan folyamává. Ha egy nyelő eléri az s-et, akkor a nyelőt és a hozzá tartozó igényt töröljük. Ismételjük ezt addig, amíg már nem lesz mozgatható nyelő. Egyszerű észrevétel, hogy ekkor minden nyelőpontot tartalmazó csúcsba legalább két él megy be.
- **Harmadik lépés.** Ez a lépés iterációkból áll: először alternáló kört keresünk, aztán megváltoztatjuk a folyamértéket a kör mentén, végül egy bizonyos szabály szerint mozgatjuk a nyelőket.

Azt mondjuk, hogy egy nyelő *reguláris*, ha a hozzá tartozó igény nagyobb, mint bármely bemenő élen a folyam értéke, egyébként *irregulárisnak* nevezzük. A második lépés után minden nyelő reguláris, hiszen az irregularitás a mozgathatóságnak felel meg. A később megadandó mozgatási szabályunk garantálni fogja, hogy ha egy csúcspontban van nyelő, akkor van reguláris nyelő is, azaz a csúcs befoka legalább kettő.

Alternáló kör keresés: Egy tetszőleges v csúcsból indulunk ki és megyünk előre, amíg tudunk, ezáltal egy *előre* utat konstruálva. Mivel a gráf aciklikus, előbb-utóbb megakadunk, és ahol megakadtunk, ott lennie kell egy t_i nyelőnek, hiszen a pont kifoka 0. Innen indulva egy *vissza* utat konstruálunk, éleken visszafelé haladva úgy, hogy nem azon az élen indulunk el, amin t_i-be jutottunk. A vissza út építését azonnal abbahagyjuk, ha egy olyan csúcsba jutunk, amiből indul ki él azon kívül is, amin oda érkeztünk. Ekkor újra egy előre utat kezdünk építeni, és így tovább, amíg egy kör nem alakul ki. Ez a kör felváltva tartalmaz előre és vissza utakat.

Folyam módosítás: Az előre utak mentén csökkenteni fogjuk a folyamértéket ϵ-nal, a vissza utak mentén pedig növelni, szintén ϵ-nal. Az így kapott folyam továbbra is egy megengedett folyam lesz, amely kielégíti az igényeket. Az ϵ értéke a következő: $\epsilon = \min(\epsilon_1, \epsilon_2)$, ahol $\epsilon_1 > 0$ az előre éleken vett folyamértékek minimuma, ϵ_2 pedig a minimuma d_j és $f(u, t_j)$ különbségének, ahol t_j reguláris nyelő, és (u, t_j) egy vissza úton szereplő él. Ha nincs ilyen pár, akkor $\epsilon_2 = \infty$. A reguláris nyelő definíciója szerint $\epsilon_2 > 0$.

Ha a minimum egy előre él mentén vétetik fel, akkor ez az él a folyamváltoztatás után eltűnik. Ha egy vissza él határozza meg a minimumot, akkor az él értéke a módosítás után d_j lesz.

Nyelők mozgatása: Azt mondjuk, hogy egy él *kényszerítetté* válik, ha vissza úton szerepel, azaz folyamot növeltünk rajta. Ha egy él egyszer kényszerített lett, akkor kényszerített marad az algoritmus végéig, vagy míg nem töröljük a gráfból. Egy t_i nyelőt a következő két esetben mozgatunk az $e = (v, t_i)$ él mentén:

1. e kényszerített, és $f(e) = d_i$, vagy
2. e nem kényszerített, és $f(e) \geq d_i$.

Ha t_i-t mozgatjuk az e mentén, akkor a folyamértéket d_i-vel csökkentjük az e-n, így ha egy kényszerített él mentén mozgatunk nyelőt, akkor az él a mozgatás után automatikusan törlődik a gráfból. Ha $\epsilon = \epsilon_2$, azaz a folyammódosítás mértékét egy visszaél határozza meg, akkor az adott él kényszerített lesz, értéke pedig pontosan d_j, az 1. szabály szerint t_j-t keresztülvisszük rajta, az él pedig eltűnik. Így minden iteráció során egy élt törlünk a gráfból.

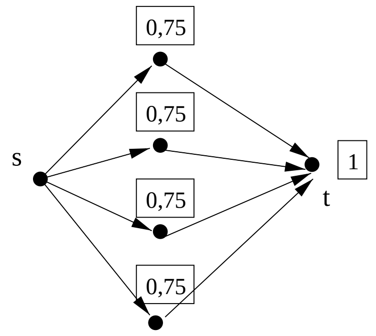
Az algoritmus véget ér, ha minden nyelő elérte az s forrást. A nyelők mozgása által meghatározott utak fogják adni az osztatlan folyamot meghatározó útvonalakat. Ha feltesszük, hogy az algoritmus működik, akkor a következő tételt kapjuk:

4.47. Tétel. *Ha létezik törtmegoldás, akkor az algoritmus olyan osztatlan folyamot határoz meg, amelyben bármely élen a folyam értéke legfeljebb* $d_{\max}$*-al haladja meg az él kapacitását (a kiindulási folyamértéket).[9]*

Bizonyítás. Mielőtt egy él kényszerítetté válna, a rajta átmozgatott nyelők összértéke legfeljebb az élen kezdetben adott folyamérték. Miután kényszerítetté válik, már csak egyetlen nyelőt mozgathatunk át rajta, azaz az osztatlan folyamban az él értéke határozottan kisebb, mint az eredeti folyamérték és a maximális igény összege. □

A fenti tétel következő erősítése is következik a bizonyításból:

4.48. Tétel. *Ha létezik törtmegoldás, akkor az algoritmus olyan osztatlan folyamot talál, hogy minden e élhez létezik egy igény, amit ha elhagyunk az e él kapacitása már nem sérül.* □



14. ábra: A 4.47. tétel szorosságát igazoló példa $q = 4$ esetben. Az igények a kis négyzetekben láthatók.

A 4.47. tétel eredménye lényegében nem javítható, amit a következő példa (14. ábra) mutat. Legyen $q \geq 2$ egész szám, a gráf álljon egy s, egy t és q darab v_i pontból. Egy kapacitású élek indulnak s-ből minden v_i-be, a v_i-kből pedig t-be. Minden v_i-n adott egy $1 - \frac{1}{q}$ nagyságú, t-n pedig egy 1 nagyságú igény. Ez a probléma törtfolyammal megoldható, és könnyen látható, hogy egy osztatlan folyam legalább $1 - \frac{1}{q}$-val megsérti az élkapacitásokat, ami – ha q nagy – körülbelül a maximális igény.

Az algoritmus futási ideje $O(mn + km)$, ahol m az élszám, n a csúcsszám. Minden iterációban ugyanis $O(n)$ lépés az alternáló kör keresés és ϵ_1 meghatározása, ez összesen $O(mn)$ lépés. Az ϵ_2 meghatározásához minden iterációban $O(k)$ lépes szükséges, ez összesen $O(km)$ lépés, hiszen az iterációk száma legfeljebb m. Nyelők mozgatására az algoritmus során $O(kn)$ lépésben kerül sor.

4.5.2. Az algoritmus helyessége

Az alábbiakban megmutatjuk, hogy az algoritmus során tett feltevéseink teljesülnek. Az alábbi állítások az iterációk kezdetére vonatkoznak.

4.49. Lemma. *Ha $e = (u, v)$ kényszerített, akkor v kifoka legfeljebb 1, és ha van kilépő él, akkor az is kényszerített.*

Bizonyítás. Mikor e kényszerítetté válik, akkor – az alternáló kör konstruálása miatt – az állítás teljesül. Ezután már csak törlünk éleket a gráfból, így az állítás az iterációk során is fennmarad. □

4.50. Lemma. *Ha v-ből indul ki él, akkor minden v-n található nyelő reguláris. Ha v-ből nem indul ki él, akkor legfeljebb egy irreguláris nyelő van v-n.*

Bizonyítás. Az algoritmus második lépése után minden nyelő reguláris. Megmutatjuk, hogy ha egy iteráció előtt az állítás teljesül, akkor az iteráció után is teljesül. Ha egy v-be érkező élen csökkentjük a folyamot, vagy nyelőt mozgatunk el v-ből, akkor nem nő a v-n található irreguláris nyelők száma. Az állításunk emiatt csak két esetben sérülhet: ha egy v-be menő élen növeljük a folyam értékét, vagy ha v-be mozgatunk nyelőket. Az első eset nem állhat elő, hiszen az alternáló kör mentén a folyammódosítás értékét (ϵ) úgy választottuk meg, hogyha v-n egy nyelő irregulárissá válna, akkor azt azonnal el tudjuk mozgatni onnan. A második eset vizsgálatához három dolgot kell meggondolni:

1. *v-ből nem indul ki él.* Ekkor nem érkezhet új nyelő v-be.

2. *v-ből legalább két él indul ki.* A 4.49. lemma szerint ekkor nem léphet be kényszerített él v-be, azaz a v-n található irreguláris nyelőket mind tovább tudjuk vinni v-ből.

3. *v-ből egy e él indul ki.* Ha v-be nem lép be kényszerített él, akkor alkalmazhatjuk a 2. eset érvelését.

   Ha v-be lép be kényszerített él, akkor e is kényszerített, azaz az első v-be érkező nyelő törli az e élet, tehát legfeljebb egy nyelő érkezhet v-be az iteráció során. Ez a nyelő akár irreguláris is lehet, ez nem baj, hiszen – indukcióval – az iteráció előtt volt kimenő él v-ből, az összes ottlévő nyelő reguláris volt, az iteráció utánra nincs kimenő él, és legfeljebb egy irreguláris nyelő lehet.

□

4.51. Állítás. *Bármely pont, amelyen vannak nyelők tartalmaz reguláris nyelőt is.*

Bizonyítás. Ha v-n legalább két nyelő van, vagy v kifoka nem nulla, akkor a 4.50. lemma szerint lennie kell reguláris nyelőnek is. Ha v-n egyetlen t_i nyelő van, és kifoka nulla, akkor a belépő éleken érkező folyam összege pontosan d_i. Ha a befok egy, akkor a belépő élen továbbvihettük volna a nyelőt, így ez nem lehetséges, ha pedig a befok legalább kettő, akkor minden belépő élen a folyamérték határozottan kisebb, mint d_i, azaz a nyelő reguláris. □

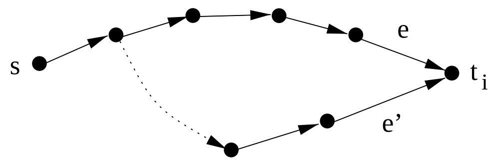
4.52. Állítás. *Bármely pont befoka legalább kettő, ha tartalmaz nyelőket.*

Bizonyítás. Tegyük fel, hogy v tartalmaz nyelőket, ekkor tartalmaz egy t_i reguláris nyelőt is a 4.51. állítás szerint. Ha a befok egy, akkor az egyetlen bemenő élen a folyamérték legalább d_i kellene, hogy legyen, ami ellentmond t_i regularitásának. □

A következő lemma megmutatja, hogy az algoritmus tényleg működik:

4.53. Lemma. *Amíg nem ért minden nyelő a forráspontra, addig az algoritmus talál alternáló kört.*

Bizonyítás. Előre út készítése során, csak olyan csúcsban akadhatunk el, amelyből nem indul ki él. Ilyen csúcs szükségszerűen tartalmaz legalább egy nyelőt, így a 4.52. állítás szerint befoka legalább kettő, azaz folytathatjuk az eljárást vissza út építésével.

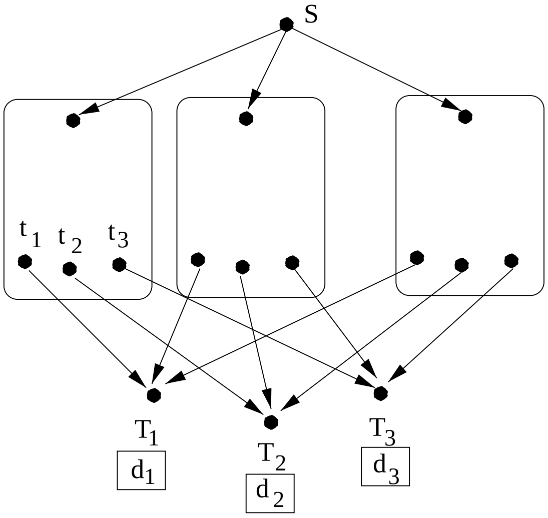


15. ábra: Visszaút építésének vége

Vissza út konstruálása akkor akad el, ha a t_i nyelőből – ahonnan indul – csupa olyan ponton át halad az út, amelynek kifoka egy, azaz nem tudunk újra előre elindulni, és végül egy nulla befokú pontba jutunk. Ez a pont csak az s lehet. A 4.52. állítás szerint kell lennie e' élnek, ami t_i-be megy, és nem egyezik meg az e éllel, ami a vissza út első éle (lásd 15. ábra). Az e'-n keresztül menő folyam meghatároz egy s-ből t_i-be menő utat. Ez az út szükségszerűen eléri egyszer a mi vissza utunkat, hiszen kezdőpontja s. Ez pedig ellentmond annak, hogy a vissza út konstruálása során csupa olyan pontot érintettünk, ahonnan nem tudunk elágazni. □

4.5.3. A fordulók számának minimalizálása

Ebben a részben azt a kérdést szeretnénk megválaszolni, hogy hány forduló szükséges az összes igény teljesítéséhez, azaz hány osztatlan folyam kell, amely nem sérti a kapacitásokat, egyenként csak az igények egy részét teljesíti, de együttesen az összes igényt kielégíti. A 4.47. tétel alkalmazásával eljárást mutatunk az igények teljesítéséhez szükséges fordulók számának minimalizálására. Feltesszük, hogy $d_{\max} \le c_{\min}$, azaz bármely igényt bármely élen keresztül tudunk vinni. Megmutatjuk, hogy ha létezik az igényeket teljesítő f törtfolyam, akkor 5 forduló elég az összes igény teljesítéséhez. Az ötlet, hogy a „kis" és „nagy" igényeket külön kezeljük. Ezek az eredmények is [9]-ben találhatók.



16. ábra: A G' segédgráf konstrukciója

4.54. Lemma. *Ha létezik törtmegoldás és $d_{\max} \le c_{\min}$, akkor az összes $(1-\frac{1}{q})d_{\max}$-nál kisebb igény elszállítható osztatlan folyammal q fordulóban.*

Bizonyítás. Készítsünk egy G' segédgráfot c' élkapacitásokkal a következő módon. Csináljunk q darab diszjunkt példányt G-ből. A G-beli e él minden példányának kapacitása legyen $c(e)/q$. Minden t_i nyelőhöz, amelyre $d_i \le$ $\le (1-\frac{1}{q})c_{\min}$ vegyünk fel egy T_i szuper-nyelőt. Kössük össze a t_i valamennyi példányát T_i-vel ∞ kapacitású élekkel. Készítsünk egy szuper-forrást is S-et, és kössük össze az s valamennyi példányával szintén ∞ kapacitású élekkel. A vizsgált „kis" igényeket helyezzük a nekik megfelelő szuper-nyelőkbe (lásd 16. ábra).

Létezik törtfolyam G'-n, amely kielégíti az igényeket, hiszen létezik egy megengedett f törtfolyam G-n, így az f/q folyam a G minden másolatában megengedett, és a q darab folyam összege teljesíti az összes igényt. A 4.47.

tétel segítségével kapunk egy f' osztatlan folyamot, amely az élkapacitásokat legfeljebb a maximális igénnyel sérti meg. Legyen $e \in E$ tetszőleges él, ekkor: $f'(e) \leq f(e)/q + (1 - 1/q))c_{\min} \leq c(e)/q + c(e)(q - 1)/q = c(e)$, azaz az eredeti $c(e)$ élkapacitás nem sérül meg egyetlen élnél sem. Ez pedig azt jelenti, hogy a G minden másolatában kaptunk egy-egy osztatlan folyamot, amely az eredeti élkapacitásokat nem sérti meg, azaz az összes „kis" igényt kielégítettük q darab osztatlan folyammal. $\square$

4.55. Lemma. *Ha létezik törtfolyam és $d_{\max} \leq c_{\min}$, akkor minden igény, amely a $[\frac{d}{q-1}, d]$ intervallumba esik, elszállítható osztatlan folyammal q fordulóban. Itt $0 \leq d \leq d_{\max}$ tetszőleges.*

Bizonyítás. Elkészítjük a G' segédgráfot ugyanúgy, mint a 4.54. lemmában, azzal a különbséggel, hogy csak azon igényeket vesszük figyelembe, amikre $d_i \in [\frac{d}{q-1}, d]$. Az algoritmus az f' osztatlan folyamot szolgáltatja. Legyen e a G egy példányának éle. Két esetet különböztetünk meg. Ha G-ben $c(e) \geq \frac{q}{q-1}d$, akkor $d \leq \frac{q-1}{q}c(e)$, és mint az előző lemmában: $f'(e) \leq \frac{c(e)}{q} + d \leq \frac{c(e)}{q} + \frac{q-1}{q}c(e) = c(e)$. Ha G-ben $c(e) < \frac{q}{q-1}d$, akkor a segédgráfban bármely igény nagyobb mint a segédgráfbeli él kapacitása, így a 4.48. tétel miatt csak egy igényhez tartozó út haladhat át e-n, így $f'(e) \leq d$. Mivel $d_{\max} \leq c_{\min}$, így az osztatlan folyam az eredeti élkapacitásokat nem sérti.

A q darab osztatlan folyamot most is a G q darab példányában kapott folyamok adják. Ezek együtt kielégítik az összes igényt. $\square$

A fenti két lemma együttes alkalmazásával kaphatjuk a következő tételt:

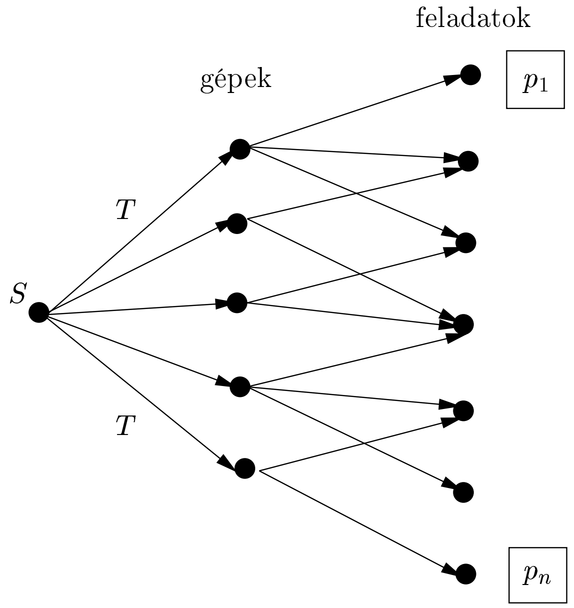
4.56. Tétel. *Ha $d_{\max} \leq d_{\min}$, és a vágási feltétel teljesül, akkor az összes igény osztatlan folyammal 5 fordulóban teljesíthető, továbbá létezik polinomiális futási idejű algoritmus, ami az 5 osztatlan folyamot megtalálja.*

Bizonyítás. Alkalmazzuk a 4.54. lemmát $q = 2$-vel, a 4.55. lemmát pedig $q = 3$-al, a kapott 5 darab osztatlan folyam minden igényt kielégít. Mivel a segédgráfnak $O(qn)$ pontja és $O(qm)$ éle van, ezért az lemmákban szereplő eljárás futási ideje $O(mn + km)$. $\square$

4.5.4. Egy ütemezéselméleti alkalmazás

Az egyforrásos osztatlan folyamok egy speciális esete a következő ütemezési feladat, amelyet Lenstra, Shmoys és Tardos [11] vizsgáltak: m darab gépen kell elvégeznünk feladatokat, azonban egy j munka csak a gépek egy $M(j)$ részhalmazán elvégezhető. A j. feladat végrehajtásának ideje p_j bármely megfelelő gépen. A cél az összes feladat elvégzésének idejét minimalizálni. Tekintsük a 17. ábrán látható folyam problémát. Az S csúcsból T kapacitású élek futnak a gépeket reprezentáló csúcsokhoz, ahonnan további élek mennek a feladatokhoz, ha az adott gépen elvégezhető az adott feladat.

Ezen élek kapacitása végtelen. Minden feladatnál található egy terminál p_i igénnyel. Ennek az egyforrásos folyam problémának pontosan akkor van osztatlan megoldása, ha a feladatok T idő alatt elvégezhetők.



17. ábra: Ütemezési probléma osztatlan folyam reprezentációja

A fenti konstrukciójú oszatatlan folyamra alkalmazva a 4.47. tételt az ütemezési feladat approximációját kapjuk:

4.57. Tétel. *Létezik polinomiális algoritmus, amely a fenti ütemezési feladatnak olyan megoldását adja, hogy a végrehajtási idő legfeljebb* $\max_j p_j$*-vel nagyobb az optimálisnál. [11]* $\square$

Irodalom

[1] N. Garg, J. Könemann *Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems*, 39th Annual IEEE Symposium on Foundations of Computer Science, pp. 300-309, 1998

[2] L. K. Fleischer *Approximating Fractional Multicommodity Flow Independent of the Number of Commodities*, 40th Annual IEEE Symposium on Foundations of Computer Science, pp. 24-31, 1999

[3] B. Shepherd, L. Zhang *A Cycle Augmentation Algorithm for Minimum Cost Multicommodity Routing on a Ring* Megjelenés alatt.

[4] D. S. Hochbaum *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, 1997.

[5] G. Wilfong, P. Winkler *Ring Routing and Wavelength Transition*, Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, Providence, RI, pp. 333-341, 1998.

[6] D. B. West *Introduction to Graph Theory*, Prentice Hall, pp. 247-257, 1996.

[7] H. Okamura, P. D. Seymour *Multicommodity Flows in Planar Graphs* J. Combin. Theory, Ser. B, Vol. 31, No.1 pp. 75-81, 1981.

[8] A. Schrijver, P. Seymour, P. Winkler *The Ring Loading Problem*, SIAM J. Discrete Math. Vol. 11, No., pp. 1-14, 1998.

[9] Y. Dinitz, N. Garg, M. X. Goemans *On the Single-Source Unsplittable Flow Problem* Combinatorica, 19, 17-41, 1999.

[10] S. Khanna *A Polynomial Time Approximation Scheme for the SONET Ring Loading Problem*, Bell Labs Techincal Journal, Spring 1997, pp. 36-41.

[11] J.K. Lenstra, D. B. Shmoys, É. Tardos *Approximation Algorithms for Scheduling Unrelated Paralell Machines,* Mathematical Programming, 46, pp. 259-271, 1990.