Alkalmazott Matematikai Lapok **35** (2018), 145–182

ERŐSEN POLINOMIÁLIS PIVOT ALGORITMUSOK A MAXIMÁLIS FOLYAM FELADATRA

ILLÉS TIBOR, MOLNÁR-SZIPAI RICHÁRD

Prékopa András és Klafszky Emil munkássága előtt tisztelegve

A dolgozatban bemutatjuk a maximális folyam feladat javítóutas algoritmusaitól eredeztethető cimkézéses technika felhasználását és továbbfejlesztését a feladat különböző polinomiális pivot algoritmus variánsainak létrehozására. Egységes rendszerben tárgyaljuk a 90-es évek primál és duál szimplex algoritmusait [6, 27] és az MBU-algoritmus variánsait [33, 34, 40], külön hangsúlyt fektetve a gyakorlati feladatok modellezése során gyakran előforduló nem nulla alsó korlátok kezelésére. Az algoritmusokat mozdony hozzárendelési feladatokon végzett számításokon hasonlítjuk össze.

1. Bevezető

A hálózati folyam modell egy felettébb népszerű optimalizálási modell, köszönhetően annak, hogy intuitív, mégis a gyakorlatban széles körben alkalmazható. Ennek megfelelően rendkívül sok publikáció foglalkozik vele, számtalan jobbnál jobb algoritmust javasolva, mind elméleti, mind gyakorlati hatékonyságukat tekintve.

A hálózati folyam természetes modellje olyan problémáknak, ahol egy rendszerben anyag áramlik bizonyos keletkezési helyektől bizonyos felhasználási helyek felé. Az előbbi helyeket forrásoknak, az utóbbiakat nyelőknek nevezzük. Anyag alatt sokféle dolgot érthetünk: lehet szó csővezetékben áramló folyadékról, utcákon mozgó járművekről, vagy számítógépes hálózaton közvetített információról. A hálózat egyes csomópontjai között az anyag szállítására alkalmas médiumok vannak (cső, utca, kábel). Ezek a legegyszerűbb modellben egyirányú áramlást engednek meg, és rendelkeznek egy kapacitással, amit az időegység alatt átvihető maximális anyag mennyiségének tekinthetünk. Folyamnak nevezzük az anyag áramlásának egy olyan leírását, ami eleget tesz ezen kapacitásoknak, és a forrásoktól és nyelőktől különböző csomópontokon csak átáramlás történik, illetve megfelelnek a forrásokból kifolyó és nyelőkbe befolyó mennyiségekre tett esetleges egyéb korlátoknak.

Egy speciális hálózati folyam feladat a maximális folyam feladat, ahol a hálózat egy forrást és egy nyelőt tartalmaz, és szeretnénk meghatározni a forrásból a nyelőbe elszállítható maximális mennyiséget, vagyis valamilyen értelemben az egész hálózat „kapacitását" (több forrás és több nyelő esetén a kapacitás keresése szintén visszavezethető erre a problémára). Megjegyezzük, hogy a fenti példákban a „nulla folyam", vagyis amikor semmi sem történik a hálózatban, egy megengedett folyam, de bonyolultabb modellek esetén már megengedett folyam keresése sem triviális.

Az egyik első hálózati folyam feladatot Tolstoi írta fel 1930-ban [51], aki egy szállítási terv optimalizálására javasolt egy módszert. Ez lényegében a negatív súlyú körök kiküszöbölésére jött rá. Logisztikai jellegű feladatok modellezésére napjainkban is használunk hálózati folyam modelleket, ilyen például a tehervontatás optimalizálása [7, 32, 44], avagy a buszközlekedésé [5]. A hálózati folyam modell tulajdonságaival, algoritmusaival és alkalmazásaival több, nagyobb lélegzetvételű mű is foglalkozik: [2, 20, 31, 37, 38, 45].

A fejezet hátralevő részében röviden összefoglaljuk a javítóutas algoritmusokat, illetve kitekintünk egyéb hatékony algoritmusokra. A 2. fejezetben ismertetjük a primál, illetve duál szimplex algoritmust maximális folyam feladaton, illetve ezek polinomiális változatait. A 3. fejezetben foglalkozunk a nem nulla alsó korlátok esetén felmerülő problémákkal, bemutatjuk az első fázis feladatot, illetve a fízibilitási MBU-algoritmust, mint primál induló megoldást előállító algoritmusokat. A 4. és 5. fejezetben ismertetjük a primál, illetve duál MBU-algoritmust, és az ezek polinomialitásának belátásához szükséges technikákat. A 6. fejezetben a mozdony hozzárendelési feladaton összehasonlítjuk az ismertetett algoritmusokat.

1.1. Javítóutas algoritmusok

Az egyik alkalmazási terület természetesen a tényleges hálózatok kapacitásának vizsgálata. A problémával foglalkozó egyik első cikkben [17] Ford és Fulkerson motivációként egy, a szovjet vasúthálózat kapacitásának felmérésével foglalkozó jelentést nevez meg [28], mely jelentés egyébként 1999-ig titkos volt [46].

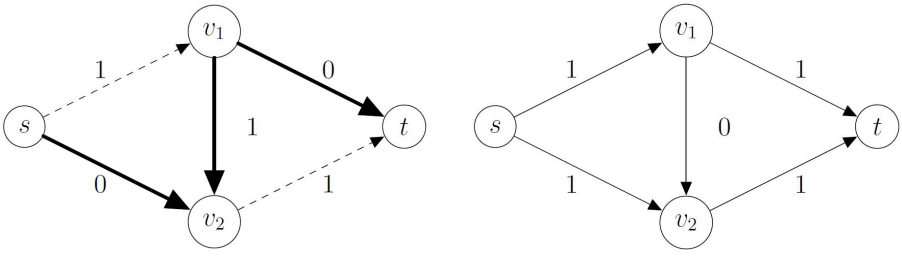
A matematikailag szabatos megfogalmazáshoz tekintsünk egy $G = (V, E)$ összefüggő, irányított gráfot egy kitüntetett $s \in V$ forrással és $t \in V$ nyelővel, valamint egy $u : E \to \mathbb{R}_{\oplus}$ kapacitásfüggvénnyel. Ekkor a feladat egy olyan $x : E \to \mathbb{R}$ függvény keresése, melyre

$$\sum_{(s,v)\in E} x(s,v) \to \max$$
$$\forall v \in V\backslash\{s,t\}: \quad \sum_{(w,v)\in E} x(w,v) = \sum_{(v,w)\in E} x(v,w)$$
$$\forall e \in E: \quad 0 \le x(e) \le u(e).$$

Itt az s-ből t-be eljutó mennyiséget a forrásból kimenő folyam mennyiségeként írtuk fel, ez a köztes csúcsokra érvényes megmaradási egyenletek miatt meg kell,

hogy egyezzen a nyelőbe érkező folyam mennyiségével. A feladat felírásából az is rögtön látszik, hogy egy lineáris programozási feladattal állunk szemben.

Természetes ötlet, hogy az $x \equiv 0$ folyamból kiindulva keressünk olyan $s \to t$ irányított utat, melyen nincsen telített él. Ekkor egy pozitív értékkel meg tudjuk növelni a folyamot ezen út mentén, ami továbbra is megengedett marad. Könnyen mutatható azonban példa olyan megengedett folyamra, ahol ilyen út nem található, de a folyamérték mégsem optimális. Tekintsük az 1a ábrát, ahol minden él kapacitása 1. Itt az első lépésben az $s \to 1 \to 2 \to t$ utat használtuk, és elakadtunk, noha a feladat optimuma nyilvánvalóan 2. Az 1b ábrán látható optimális megoldást ekkor úgy kaphatjuk meg, hogy az $s \to 2 \leftarrow 1 \to t$ úton „javítunk” 1-et, ahol javítás alatt az előreéleken növelést, míg a visszaéleken csökkentést értünk.



(a) egy javítóút (b) a javítás után

1. ábra. Javítóutas módszer

Javítóút alatt tehát egy olyan s-ből t-be menő P irányítatlan utat értünk, melynek előreéleire $x < u$, visszaéleire pedig $x > 0$. Ekkor az úton

$$\delta = \min_{e \in P} \left\{ \begin{array}{cc} u(e) - x(e), & \text{ha } e \text{ előreél,} \\ x(e), & \text{ha } e \text{ visszaél} \end{array} \right\} > 0$$

mennyiséget tudunk javítani.

A javítóút már tényleg megfelelő objektum a folyam optimalitásának vizsgálatára:

1.1. Lemma. *Az x megengedett folyam pontosan akkor optimális, ha nem található a gráfban javítóút.*

Bizonyítás vázlat: Optimális folyamhoz természetesen nem létezhet javítóút.

A másik irány belátásához tekintsük azon csúcsok S halmazát, melyek elérhetők s-ből javítóút segítségével. Ekkor belátható, hogy az S-ből $V \backslash S$-be menő élek telítettek, és értékük megegyezik a folyam értékével, tehát az $(S, V \backslash S)$ vágáson nem lehet több folyamot átvinni, mint amennyit x átvisz, így x-nek maximálisnak kell lennie. □

A bizonyítás gondolatából kijön továbbá Ford és Fulkerson híres maximális folyam–minimális vágás tétele is:

1.1. Tétel. (Ford, Fulkerson [18]) *Az s-ből t-be szállítható maximális folyam értéke megegyezik az s és t csúcsokat elválasztó vágások minimális értékével.*

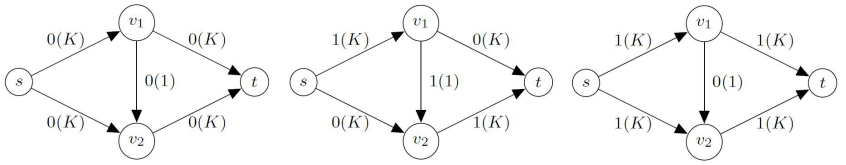
A tétel elméleti jelentőségén túl egy olyan, a feladat méretéhez viszonyítva kis méretű „tanú” létezését garantálja, mellyel nagy feladatok esetén is könnyen ellenőrizhető az adott folyam optimalitása.

Ford és Fulkerson eredményeiből látszik, hogy az általuk felírt javítóutas algoritmus (1. algoritmus) korrekt eredményt ad, ha megáll. Egész kapacitásokkal rendelkező folyam esetén a talált javítóúton mindig legalább 1-et tudunk javítani, így az algoritmus tényleg megáll, viszont irracionális kapacitások mellett az algoritmus végessége nem garantált. Erre Ford és Fulkerson könyve [19] is hoz egy példát, de a legkisebb ilyen hálózat mindössze 6 csúcsot és 8 élt tartalmaz [52].

1. Algoritmus. Javítóutas algoritmus

```
1. Ford–Fulkerson(G, c, s, t)
2.     Legyen x egy megengedett folyam          ▷ például x = 0
3.     amíg találunk javítóutat s-ből t-be,
4.         javítsunk az úton
5.     vége
6.     Megállunk, x optimális folyam.
7. vége
```

Azonban egész kapacitásokkal rendelkező hálózat esetén sem kielégítő az algoritmus. A fenti gondolatból adódó elméleti futásidő $\mathcal{O}(|E|K)$, ahol K a legnagyobb kapacitás a hálózatban. Tekintsük például a korábban látott egyszerű hálózatot, némileg megnövelt kapacitásokkal (2a ábra).



(a) indulás (b) első javítás után (c) második javítás után

2. ábra. Ford–Fulkerson-algoritmus szerencsétlen útválasztással.

Itt első javítóútnak az $s \to 1 \to 2 \to t$ utat választva 1-et tudunk javítani az $(1, 2)$ él kapacitása miatt (2b ábra). Majd az $s \to 2 \leftarrow 1 \to t$ javítóutat választva ismét csak 1-et tudunk javítani (2c ábra). Ezt a lépéspárt K-szor ismételve eljutunk a maximális folyamhoz, ahol K egység folyik az $s \to 1 \to t$ és az $s \to 2 \to t$ utakon.

1.2. Polinomiális javítóutas algoritmusok

Az előző példát (2. ábra) tehát 2 javítóúttal is meg lehet oldani, amik ráadásul rövidebbek is, mint a példában használt utak. Ez azért is érdekes, mivel az algoritmus egy kézenfekvő implementációjában szélességi kereséssel keresve javítóutat mindig egy lehető legrövidebb javítóutat használunk, így az előző példát hatékonyan oldjuk meg.

Az 1970-es évek elején Edmonds és Karp [16], illetve tőlük függetlenül Dinic [15] is belátta, hogy ez nem csak kényelmes, de hatékonyabb is:

1.2. Tétel. (Edmonds, Karp [16]; Dinic [15]) *Ford és Fulkerson algoritmusa (1. algoritmus) minden lépésben egy lehető legrövidebb javítóutat használva legfeljebb* $\frac{1}{2}|V| \cdot |E|$ *javítás után véget ér.*

A tétel bizonyításának fő felismerése, hogy a csúcsoknak a forrástól vett, javítóúton való távolsága az algoritmus során nem csökkenhet, ha mindig egy legrövidebb javítóút mentén javítunk. Ezt a távolságot szokták a v csúcs $d(v)$ „címkéjének" is nevezni, ami így az algoritmus során monoton növekszik.

Ha az (i, j) él az általunk használt javítóúton fekszik, akkor $d(i) + 1 = d(j)$, mivel egy legrövidebb javítóutat használunk. Ha ez az él telítődik a javítás során, akkor legközelebb csak visszaélként használhatjuk, vagyis ekkor $d'(j) + 1 = d'(i)$ fog teljesülni. A j csúcs címkéjének monoton növekedéséből így azt kapjuk, hogy $d'(i) \geq d(i) + 2$. Mivel egy csúcs címkéje legfeljebb $|V| - 1$ lehet, vagy végtelen, így az él legfeljebb $\frac{1}{2}|V|$-szer telítődhet, és hasonló érveléssel $\frac{1}{2}|V|$-szer csökkenhet 0-ra. Mivel minden javításnál van legalább egy ilyen él, így legfeljebb $|V| \cdot |E|$ javítás történhet. Több munkával (például figyelembe véve a nyelőtől való távolságot is) bizonyítható az $\frac{1}{2}|V| \cdot |E|$ korlát is.

A címkézési technikát és a bizonyítás gondolatmenetét később többen is felhasználták, így érdemes összefoglalni, hogy milyen lépésekből áll:

- A csúcsok egy korlátos címkézésének bevezetése.
- A címkék monotonitásának megmutatása.
- Annak megmutatása, hogy címkék rendszeresen növekednek is.
- Korlát adása a lépések számára.

A legrövidebb javítóutas algoritmus tehát erősen polinomiális. Naivan implementálva minden javításhoz szükségünk van egy javítóút megkeresésére ($\mathcal{O}(|E|)$ lépés), illetve a kapott, legfeljebb $|V|$ hosszú úton a javítás elvégzésére ($\mathcal{O}(|V|)$ lépés). Így az algoritmus futásideje $\mathcal{O}(|V| \cdot |E|^2)$.

Itt tehát a javítóutak keresgélése viszi el az idő nagy részét. Dinic a javítások számának megbecsülése mellett egy ügyesebb implementációt is javasolt, lényegében az összes k hosszú javítóutat egyszerre keresi meg. Egy $\mathcal{O}(|E|)$ lépésszámú szélességi kereséssel felépíthető egy szintezett gráf, amelyen $\mathcal{O}(|V| \cdot |E|)$ lépéssel

talál egy blokkoló folyamot. A blokkoló folyam eredeti hálózathoz adása után a legrövidebb javítóút hossza legalább 1-gyel nő, így ilyen segédfeladatból kevesebb, mint $|V|$ darabot kell megoldani, így $\mathcal{O}(|V|^2|E|)$-re csökken az algoritmus futásideje.

1.3. Egyéb algoritmusok

Új megközelítést jelentett az úgynevezett előfolyamokra épülő módszer. Az előfolyam a folyamnál gyengébb struktúra: ugyanúgy megköveteljük a kapacitások betartását, de a közbülső csúcsokra csak annyit kötünk ki, hogy a bejövő mennyiség legalább annyi legyen, mint a kimenő. A módszer a javítóutas algoritmusokkal ellentétes irányból közelíti meg a problémát. Míg ott a folyam struktúrát minden lépésben megőrizve növeltük a folyam értékét a maximumig, addig az előfolyamos algoritmusoknál a forrásból kiáramló mennyiség optimális, vagy a fölötti szinten tartása mellett lépkedünk előfolyamokon, amíg a struktúra meg nem javul.

Karzanov 1974-es cikke [36] számít a módszer alapművének. A Dinic által bevezetett részfeladatokat gyorsabban megoldva $\mathcal{O}(n^3)$ lépésszámú algoritmust sikerült létrehoznia. Így minden fázis után egy valódi folyamot kapunk.

Későbbi előfolyam algoritmusoknál, egészében kezelve a feladatot, csak az algoritmus végén kapunk valódi folyamot. Különösen érdekes számunkra Goldberg és Tarján előfolyam algoritmusa [24, 25]. Ez a csúcsoktól a nyelőig vezető javítóút távolságát becslő címkézést használ. Minden lépésben egy kisebb címkéjű csúcs felé tol folyamot az algoritmus, illetve ha egy aktív csúcsnak (ahol több a bejövő folyam, mint a kimenő) nincs ilyen szomszédja, akkor újracímkézi azt. Bebizonyítható, hogy némi rendszerezettséggel (például FIFO-szabállyal választva az aktív csúcsok közül) legfeljebb $\mathcal{O}(n^2)$ újracímkézés és $\mathcal{O}(n^3)$ tolás történik az algoritmusban. Mivel egy tolás műveletigénye konstans, szemben a javítóutas algoritmusoknál használt $\mathcal{O}(n)$ hosszú úton történő javítással, így az algoritmus teljes műveletigénye is $\mathcal{O}(n^3)$. A két jellemző művelet alapján „push-relabel" algoritmusként is hivatkozzák az ehhez hasonló algoritmusokat az angol nyelvű szakirodalomban.

Ezek az előfolyamos algoritmusok már nemcsak az elméleti lépésszám, de különböző tesztfeladatokon mért futásidő tekintetében is jobban teljesítettek mint a korábbi javítóutas algoritmusok [1, 3, 14]. Itt jegyeznénk meg, hogy az elméleti lépésszám tovább javítható speciális adatstruktúrák használatával [22, 47, 48], de a gyakorlatban ez sokszor az algoritmus lassulásához vezet.

Szintén kombinatorikus megközelítés eredménye Hochbaum úgynevezett *pszeudófolyam* algoritmusa [29]. Ez az előfolyam algoritmusnál általánosabb módon megengedi a közbülső csúcsokra a megmaradási egyenletek bármelyik irányban történő megsértését. Az induló megoldás folyamértéke legalább a maximális folyam értéke, majd a többlettel rendelkező csúcsokból a szükséglettel rendelkező csúcsok felé terelve próbálja a pszeudófolyamot szokásos folyammá alakítani a folyam értékét nem csökkentve. Amikor ez már nem lehetséges, akkor a pszeudófolyam egyszerűen visszaalakítható egy maximális folyammá. Az algoritmus az

előfolyam algoritmusoknál használt technikákkal polinomiálissá tehető, továbbá létezik egy pivotalgoritmus variánsa is.

2. Pivot algoritmusok

A maximális folyam feladat, és általánosabban a minimális költségű hálózati folyam feladat felfogható lineáris programozási feladatként is. Vezessünk be egy (t,s) élt, amin az összes nyelőbe érkező folyamot visszavezetjük a forrásba, $E^* := E \cup (t,s)$ és $G^* := (V, E^*)$. Ekkor a csúcsokra vonatkozó megmaradási egyenletek egységesebbé válnak, és a célfüggvény felírható a (t,s) élen folyó folyam maximalizálásaként:

$$\begin{aligned} & \max x_{t,s} \\ \forall v \in V: \quad & \sum_{(w,v)\in E^*} x_{w,v} - \sum_{(v,w)\in E^*} x_{v,w} = 0 \\ \forall e \in E: \quad & l_e \le x_e \le u_e. \end{aligned} \tag{1}$$

A csúcsokra vonatkozó egyenletek röviden $A\mathbf{x} = \mathbf{0}$ alakba írhatók, ahol A a G^* gráf illeszkedési mátrixa. Ismert, hogy egy n csúcsú összefüggő gráf illeszkedési mátrixának rangja $n-1$, továbbá a mátrix valamely $n-1$ oszlopa pontosan akkor lineárisan független, ha az oszlopoknak megfelelő élek feszítőfát alkotnak a gráfban.

Következésképpen (1) egy B bázisa megfelel G^* egy T feszítőfájának, a fán kívüli élek folyamértéke megegyezik az alsó vagy felső korlátjukkal, ebből is következik, hogy T mindig tartalmazza a (t,s) élt. Így egy feszítőfához több bázismegoldás is tartozik, szokás a változókat három részre osztani: (B, L, U), ahol B a bázisban levő változók, L az alsó korláton, U pedig a felső korláton levő bázison kívüli változók.

Irányított gráf illeszkedési mátrixáról ismert továbbá, hogy teljesen unimoduláris, vagyis minden négyzetes részmátrixának determinánsa 0, 1 vagy -1. Ebből egész jobboldal esetén következik (például a Cramer-szabály alapján), hogy a feladat bármely bázismegoldása egészértékű, ami bizonyos alkalmazásoknál feletteb hasznos tulajdonság.

Primál megengedett bázismegoldásnak egy olyan bázismegoldást nevezünk, ahol az $l_e \le x_e \le u_e$ korlátok is teljesülnek. A bázison kívüli változókra ez automatikusan teljesül, vagyis primál nem megengedett változó csak a bázisban lehet.

A duál megengedett bázismegoldások karakterizálásához hagyjuk el a (t,s) élt a T feszítőfából. Ekkor a feszítőfa két összefüggő részre esik, jelöljük S-sel az s csúcsot tartalmazó részfát, és Z-vel a t csúcsot tartalmazót. Egy *bázismegoldás duál megengedett*, ha az $S \to Z$ élekre $x_e = u_e$, míg a $Z \to S$ élekre $x_e = l_e$

teljesül. Ennek belátásához írjuk fel az (1) feladat duálisát:

$$\begin{aligned} & \min \sum_{e\in E} u_e\lambda_e - l_e\mu_e \\ \forall (v,w)\in E: \quad & \pi(w)-\pi(v)+\lambda_{v,w}-\mu_{v,w}\geq 0 \\ & \pi(s)-\pi(t)\geq 1 \\ \forall e\in E: \quad & \lambda_e,\mu_e\geq 0. \end{aligned}$$

Elég meggondolnunk, hogy a duális feladat következő megoldása megengedett, valamint komplementáris a primál feladat fenti struktúrájú megoldásának eltérésváltozóira (λ az $\mathbf{u}-\mathbf{x}$ eltérésekre, μ pedig az $\mathbf{x}-\mathbf{l}$ eltérésekre):

$$\pi(v)=\begin{cases} 1, & \text{ha } v\in S,\\ 0 & \text{egyébként.}\end{cases}$$

$$\lambda_{v,w}=\begin{cases} 1, & \text{ha } v\in S \text{ és } w\in Z,\\ 0 & \text{egyébként.}\end{cases}$$

$$\mu_{v,w}=\begin{cases} 1, & \text{ha } v\in Z \text{ és } w\in S,\\ 0 & \text{egyébként.}\end{cases}$$

Így egy adott T feszítőfához tartozó bázismegoldásban a duál nem megengedett változók azon $S\to Z$ élekhez tartozó változók, melyekre $x_e=l_e$, és azon $Z\to S$ élekhez tartozó változók, melyekre $x_e=u_e$. Optimális megoldás esetén az (S,Z) vágás egy a Ford–Fulkerson-tétel által garantált minimális vágás.

2.1. Primál és duál szimplex módszer

A lineáris programozási feladat pivot algoritmussal történő megoldása során bázismegoldásról „szomszédos" bázismegoldásra lépünk, egy ilyen lépést nevezünk pivotálásnak. Ehhez ki kell választanunk egy, a bázisba belépő, és egy onnan kilépő változót úgy, hogy a csere után is bázismegoldásunk legyen. Ez folyam problémák esetén a feszítőfa struktúra megtartását jelenti.

Primál szimplex módszer esetén egy primál megengedett bázismegoldásból indulunk, egy duál nem megengedett változó lép be a bázisba, és a primál hányadosteszt segítségével választott változó lép ki a bázisból, így egy primál megengedett bázismegoldást kapunk, mely célfüggvényértéke nem romlott. Ezt a lépést addig ismételjük, amíg duál megengedett, és egyben optimális megoldást nem kapunk (vagy belátjuk, hogy a célfüggvény nem korlátos). Ez véges sok lépés után megtörténik, amennyiben teszünk valamit a ciklizálás elkerülése érdekében. Erre természetesen működnek az általános lineáris programozási feladatoknál alkalmazott szabályok

(Bland-szabály [9], lexikografikus szabály [10], s-monoton szabályok [11],...), de léteznek speciálisan hálózati folyamokra kifejlesztett megoldások is [12, 39].

Maximális folyam feladat esetében egy T feszítőfához tartozó primál megengedett bázismegoldással kezdünk (ilyen megoldás keresésének nehézségeit a 3. fejezetben részletezzük). Kiválasztunk egy p duál nem megengedett változót, ez szükségszerűen az (S, Z) vágás egy éle. A $T \cup \{p\}$ élhalmaz tartalmaz egy p-n és (t, s)-en is átmenő P kört. A kilépő élt ebből a körből kell választanunk, hogy helyreálljon a bázis struktúra (pivot táblán p oszlopában pontosan a kör éleinek megfelelő változók együtthatója nem nulla). A primál hányadostesztnek ebben az esetben megfelel a javítóutaknál látott δ kiszámítása: tekintsük a P kört a (t, s) éllel egyező irányítással, ekkor legyen

$$\delta = \min_{e \in P} \left\{ \begin{array}{ll} u_e - x_e, & \text{ha } e \text{ előreél,} \\ x_e - l_e, & \text{ha } e \text{ visszaél} \end{array} \right\} \geq 0.$$

P előreélein δ-t növelve, visszaélein δ-t csökkentve továbbra is megengedett folyamot kapunk. Egy olyan q élt kivéve a bázisból, ahol δ felvette a minimumát, pedig egy hozzá tartozó feszítőfát, hiszen x_q értéke ekkor u_e vagy l_e lesz. Az új feszítőfa tehát $T' = T \cup \{p\} \backslash \{q\}$. A folyam értéke, vagyis $x_{t,s}$ értéke δ-val növekedett, ami a javítóutas algoritmusokkal ellentétben lehet 0 is (degenerált pivot).

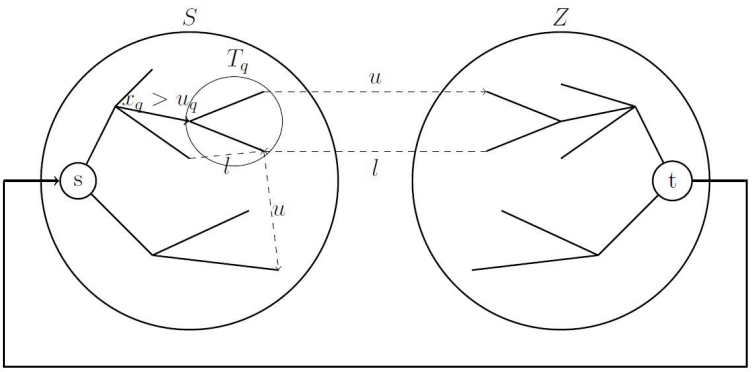
2. Algoritmus. Hálózati primál szimplex algoritmus

1. Legyen x egy megengedett bázismegoldás.
2. **amíg** x nem duál megengedett
3. p tetszőleges duál nem megengedett él.
4. $P := T \cup \{p\}$-ben található kör, (t, s)-sel egyező irányítással.
5. Meghatározzuk δ-t és q-t.
6. Módosítjuk a folyamértékeket P-n.
7. Az új bázis $T \cup \{p\} \backslash \{q\}$.
8. **vége**

Duál szimplex módszer esetén duál megengedett bázismegoldásból indulunk, egy primál nem megengedett változó lép ki a bázisból, és duál hányadosteszt segítségével választjuk ki a belépő változót. Így duál megengedett bázismegoldásokon haladunk, amíg az primál megengedett nem lesz (vagy belátjuk, hogy a feladatnak nincs primál megengedett megoldása). Ciklizálás elleni szabályok segítségével itt is biztosítható a véges futásidő.

Maximális folyam feladat esetében egy T feszítőfához tartozó duál megengedett bázismegoldással kezdünk. Kiválasztunk egy q primál nem megengedett élt. q elhagyásával T két részre esik, legyen T_q a (t, s) élt nem tartalmazó részfa. Ekkor a belépő élnek össze kell kötnie T_q-t $T \backslash T_q$-val, továbbá lehetővé kell tennie, hogy

q-t a megfelelő irányban módosíthassuk. Például tekintsük azt az esetet, amikor $x_q > u_q$, és q a $T\backslash T_q$ részfából a T_q részfába megy. Ekkor a lehetséges belépő élek a $T_q \to T\backslash T_q$ irányú $x = u$ folyamértékű élek, illetve a $T\backslash T_q \to T_q$ irányú $x = l$ élek (lásd 3. ábra). Ezek közül duál hányadosteszttel választunk: az (S, Z) vágásbeli élekre ez 1, míg a többire 0, így ez utóbbiakat preferáljuk. A belépő él kiválasztása után a kialakuló körön annyit javítunk, hogy a kilépő él folyamértéke a megfelelő korláttal egyezzen meg ($x_q > u_q$ esetén u_q-val, $x_q < l_q$ esetén l_q-val).



3. ábra. Lehetséges belépő élek a duál szimplex algoritmusban.

Amennyiben nem találunk belépő élt, úgy a feladatnak nincs megengedett megoldása. Ez nulla alsó korlátú maximális folyam feladat esetén nem lehetséges, így ez a helyzet ott nem fordulhat elő. Nemnulla alsó korlát esetén ez a helyzet előállhat, a részleteket a 3. fejezetben tárgyaljuk.

3. Algoritmus. Hálózati duál szimplex algoritmus

1. Legyen x egy duál megengedett bázismegoldás.
2. **amíg** x nem primál megengedett
3. q tetszőleges primál nem megengedett él.
4. Legyen P a lehetséges belépő élek halmaza.
5. **ha** $P \subseteq (S, Z)$ **akkor**
6. Legyen $p \in P$ tetszőleges.
7. **egyébként**
8. Legyen $p \in P\backslash(S, Z)$ tetszőleges.
9. **vége**
10. Módosítjuk a folyamértékeket a kialakuló körön.
11. Az új bázis $T \cup \{p\}\backslash\{q\}$.
12. **vége**

2.2. Polinomiális primál szimplex módszer maximális folyam feladatra

Bár a primál szimplex módszer fent leírt szemléletes értelmezése tulajdonképpen egyidős a szimplex módszerrel (már Dantzig 1963-as lineáris programozási könyvében is leírásra került [13]), erősen polinomiális változatot csak 1990-re publikáltak belőle.

Átnézve a primál szimplex algoritmus pszeudókódját (2. algoritmus), láthatjuk, hogy számottevő döntési lehetőségünk a belépő él kiválasztásánál van. A kilépő él lényegében egyértelmű, ha a felső korlátok kellően változatosak, így a továbbiakban sem vezetünk be szabályt arra nézve, hogy több lehetséges kilépő él esetén melyiket választjuk.

Goldfarb és Hao eredménye [26, 27] tényleg egy, a belépő él kiválasztására adott szabály, lényegében a legrövidebb javítóutas algoritmus során alkalmazott „címkézés" (forrástól való távolság a redukált gráfban) hozzáigazítása a szimplex módszer bázis struktúrájához. Megjegyezzük, hogy a fejezet további részében 0 alsó korlátokat feltételezünk, az algoritmus értelemszerűen módosítható alsó korlátos feladatra, plusz munkát csak egy kiinduló primál megengedett bázis megoldás keresése jelent (amiről többet a 3. fejezetben szólunk).

2.1. Definíció. (pszeudo-javítóút) Legyen az i. pivotálás előtti megengedett bázismegoldásunk x^i, a T^i feszítőfa mellett. Ekkor egy v csúcsból w csúcsba vezető élsorozat pszeudo-javítóút, ha a következő típusú élekből áll:

- T^i-n kívüli előreél $x^i_e = 0$ folyamértékkel,
- T^i-n kívüli visszaél $x^i_e = u_e$ folyamértékkel,
- $T^i \setminus \{(t,s)\}$-beli él tetszőleges irányban.

Láthatjuk, hogy a pszeudo-javítóút a javítóút fogalmának bővítése a bázisbeli élek tetszőleges használatával.

2.2. Definíció. (címkézés) Egy v csúcs címkéje az i. pivotálás előtt a legrövidebb s-ből v-be menő pszeudójavítóút hossza.

Ezen címkék kiszámítása egy egyszerű szélességi kereséssel történhet (4. algoritmus).

4. Algoritmus. Goldfarb–Hao-címkézés

1. $d_i(s) = 0$, lista $= \{s\}$
2. **amíg** lista $\neq \emptyset$
3. v legyen a lista első eleme
4. **minden** $(v,w) \in E$, $((v,w) \in T_i$ vagy $x_{v,w} = 0)$, w címkézetlen
5. $d_i(w) = d_i(v) + 1$
6. w a lista végére kerül

vége
minden $(w,v) \in E$, ($(w,v) \in T_i$ vagy $x_{w,v} = u_{w,v}$), w címkézetlen
    $d_i(w) = d_i(v) + 1$,
    w a lista végére kerül
vége
töröljük v-t a listáról
vége

Az i. pivotálás során lehetséges belépő élek C^i halmaza az (S, Z) vágás megfelelő élei (S-ből Z-be menő $x_e^i = 0$, illetve Z-ből S-be menő $x_e^i = u_e$ folyamértékű élek), ezek közül egy olyat választunk ki, melynek S-beli végpontjának címkéje minimális (5. algoritmus).

5. Algoritmus. Címkézéses primál szimplex (Goldfarb, Hao)

T^1 tetszőleges feszítőfa, $x^1 \equiv 0$, $i = 1$.
Goldfarb–Hao-címkézés
amíg $d_i(t) \neq \infty$
    válasszunk belépő élt: egy minimális S-beli címkéjű él
    válasszunk kilépő élt: primál hányados teszt
    végezzük el a pivotálást, $i = i + 1$.
    Goldfarb–Hao-címkézés
vége

Megjegyzés: a címkézést nem feltétlenül kell minden pivotálásnál újraszámolni [27]. Ez némileg javít az implementáció műveletigényén, de pivotálások számán nem, így a részletekbe nem megyünk bele.

Az így pontosított algoritmus már erősen polinomiális lesz:

2.1. Tétel. (Goldfarb, Hao [27]) *Goldfarb és Hao algoritmusa (5. algoritmus) legfeljebb nm pivotálással megtalálja a maximális folyam feladat egy optimális bázismegoldását.*

A részletes bizonyítás megtalálható az eredeti cikkben. Vázlatosan a legrövidebb javítóutas algoritmusoknál ismertetett sémát követi: belátja a címkék monotonitását, illetve rendszeresen bekövetkező valódi növekedését; ebből és a címkézés korlátosságából felső becslést ad a pivotálások maximális számára. Mivel a monotonitási lemma különböző változatai még többször elő fognak kerülni, így ennek a részletes bizonyítását ismertetjük.

2.1. Lemma. (Goldfarb, Hao [27]) *Goldfarb és Hao algoritmusa (5. algoritmus) során a csúcsok címkéi monoton növekednek, vagyis minden $z \in V$-re és minden i-re $d_i(z) \leq d_{i+1}(z)$.*

Bizonyítás: Legyen az i. pivotálásnál belépő él $p = (v, w)$, és tegyük fel, hogy $x_p^i = 0$ (vagyis $v \in S_i$ és $w \in Z_i$). Az $x_p^i = u_p$ eset hasonlóan bizonyítható.

Figyeljük meg, hogy $d_i(w) = d_i(v) + 1$. Egyrészt a címkézés definíciója alapján $d_i(w) \leq d_i(v) + 1$. Másrészt ha $d_i(w) \leq d_i(v)$ lenne, akkor a legrövidebb s-ből w-be vezető pszeudójavítóút első (S, Z) vágásbeli éle lehetséges belépő él, és kisebb S-beli címkével rendelkezne, mint p, evvel ellentmondva p választásának.

Ha az s-ből z-be vezető pszeudo-javítóút hossza csökkent, akkor létezik olyan él, amit a pivotálás után olyan irányból tudunk használni, ahogyan a pivotálás előtt nem tudtuk. Vizsgáljuk meg tehát, hogy hogyan változik az élek „használhatósága".

A $T_i \cap T_{i+1}$-beli éleket mindkét címkézésnél mindkét irányban lehet használni, a $\overline{T_i \cup T_{i+1}}$ éleknek pedig nem változott a folyamértéke, így mindkét címkézésnél ugyanabban az irányban lehet őket használni.

Ha a kilépő él is p, akkor p-t az i. címkék meghatározásánál csak előreélként, míg az $i + 1$. címkék meghatározásánál csak hátraélként lehet használni. Vagyis z címkéje csak akkor csökkenhetett, ha az $i + 1$. pivotálás előtti állapot szerinti legrövidebb s-ből z-be menő pszeudójavítóút visszaélként használja p-t.

Ha a kilépő él $q \neq p$, akkor p-t az i. címkék meghatározásánál csak előre irányban, míg az $i + 1$. címkékhez mindkét irányban használhatjuk. A kilépő q élt pedig az i. címkék meghatározásnál mindkét irányban, míg az $i + 1$. címkékhez csak az egyik irányban tudjuk használni. Tehát ebben az esetben is csak úgy csökkenhet z címkéje, ha az új legrövidebb pszeudójavítóút visszaélként használja p-t.

De ekkor a következő egyenlőtlenség-láncnak kéne fennálnia:

$$\begin{aligned} d_{i+1}(z) = d_{i+1}(w) + 1 + d_{i+1}(v, z) &\geq d_i(w) + 1 + d_i(v, z) \\ &= d_i(v) + 2 + d_i(v, z) \geq d_i(z) + 2, \end{aligned}$$

ahol $d_j(v, z)$ a legrövidebb x^j melletti $v \to z$ pszeudo-javítóút hossza. A lépések indoklásai rendre:

1. Megállapítottuk, hogy a legrövidebb x^{i+1} melletti $s \to z$ pszeudo-javítóút $s \to w \xrightarrow{p} v \to z$ alakú.
2. A legrövidebb x^{i+1} melletti $s \to w$ és $v \to z$ pszeudó-javítóutak nem használhatják visszafele a (v, w) élt, így ezek az utak legalább olyan hosszúak, mint x^i melletti párjuk.
3. Itt felhasználjuk, hogy $d_i(w) = d_i(v) + 1$.
4. Háromszög egyenlőtlenség: $d_i(z) \leq d_i(v) + d_i(v, z)$.

Vagyis ebben az esetben sem csökkent z címkéje. □

Az algoritmus egyszerű adatstruktúrákkal implementálható $\mathcal{O}(nm^2)$ műveletigénnyel, illetve az újracímkézés szükségességére figyelve $\mathcal{O}(n^2m)$ műveletigénnyel.

Az úgynevezett dinamikus fa adatstruktúrát [48] alkalmazva pedig a futásidő levihető $\mathcal{O}(nm \log n)$-re [23], ami már egy logaritmikus faktortól eltekintve megegyezik az előfolyamos algoritmusok lépésszámával.

2.3. Polinomiális duál szimplex módszer maximális folyam feladatra

Az első maximális folyam feladaton erősen polinomiális duál szimplex változat Orlintól származik [41]. Ez az algoritmus az általánosabb minimális költségű hálózati folyamra adott hatékony algoritmust. Ebben a fejezetben egy későbbi, de egyszerűbb algoritmust mutatunk be az Armstrong, Chen, Goldfarb és Jin szerzőktől [6], amely szintén a csúcsok egy címkézésével biztosítja a polinomialitást.

Az egyszerűbb tárgyalás érdekében bázisfolyamok helyett bázis előfolyamokkal dolgozunk. Egy $\mathbf{x} : E \to \mathbb{R}$ függvény *előfolyam*, ha minden élre $0 \leq x_e \leq u_e$ teljesül, illetve minden $v \in V \backslash \{s\}$ csúcsra a csúcs $e(v) := \sum_{(w,v)\in E} x_{w,v} - \sum_{(v,w)\in E} x_{v,w}$ *többlete* nemnegatív. Egy csúcsot *aktív*nak nevezünk, ha $e(v) > 0$. Az előfolyam bázis előfolyam, ha minden $e \notin T$ élre $x_e = 0$, vagy $x_e = u_e$.

Minden bázis előfolyam visszaalakítható egy nem feltétlenül primál megengedett bázis folyammá az aktív csúcsokban levő többletek forrásba való visszatolásával (a feszítőfában levő egyértelmű $s \to v$ úton csökkentjük a folyamértéket $e(v)$-vel). Ez fordítva nem igaz, hiszen egy tetszőleges nem megengedett bázisfolyam hasonló átalakításánál kialakulhatnak $e(v) < 0$ csúcsok is.

Az algoritmus során a T feszítőfát s gyökérrel fogjuk elképzelni. Az s csúcsot legfelülre rakva (mint a valós fáknál szokás) mindegyik feszítőfabeli él „felfelé" vagy „lefelé" mutató él. Az algoritmus során többletet mindig a forrás felé, felfelé fogunk tolni, egy bázisbeli él „felfelé mutató reziduális kapacitásán" felfelé élek esetén az $u_e - x_e$, míg lefelé élek esetén az x_e értéket értjük. T_v-vel jelöljük a feszítőfa v gyökerű részfáját (de megtartva a $Z = T_t$ és S jelöléseket is), illetve $pred(v)$-vel jelöljük a $v \neq s$ csúcs fölötti bázisbeli élt (a T-beli $v \to s$ út első élét).

Az algoritmus során duál megengedett bázis előfolyamokon haladunk. Kezdő megoldásnak megfelel a következő: T legyen egy $S = \{s\}$ és $Z = V \backslash \{s\}$ részekkel rendelkező feszítőfa, illetve legyen minden $(s, v) \in E$ élre $x_{s,v} = u_{s,v}$, majd az így kialakuló többletekből toljunk fel a fán annyit, amennyit lehet.

Duál szimplex algoritmusként először a q kilépő élt választjuk ki. Ez $q = pred(v)$ lesz, egy tetszőleges v aktív csúcsra ($e(v) > 0$). Ezután a belépő él a T_q részfából kimenő, reziduális kapacitással rendelkező él kell, hogy legyen. Ezek közül egy címkézési technikával választunk.

Legyen $A_{T,x} = \{(v, w) : (v, w) \in T$ vagy $(w, v) \in T$, vagy $x_{v,w} = 0$, vagy $x_{w,v} = u_{w,v}\}$, és $\widehat{A}_{T,x} = A_{T_x} \backslash \{(s, t)\}$. Egy $d : V \to \mathbb{N}$ címkézést *helyes címkézésnek* nevezünk, ha $d(t) = 0$, $d(s) = n$, és minden $(v, w) \in A_{T,x}$ esetén $d(v) \leq d(w) + 1$. Ekkor könnyen látható, hogy $d(u)$ egy alsó becslés a legrövidebb $u \to t$ irányított út hosszára $A_{T,x}$-ben, ha az (s, t) él hossza n, és minden más él hossza 1. Az algoritmus végig egy helyes címkézéssel dolgozik, a fenti legrövi-

debb utakkal kapott „egzakt” címkézés helyett, így az implementáció lépésszáma javul, az algoritmus helyességének bizonyítása viszont némileg bonyolultabbá válik. Ezzel a címkézéssel olyan élek belépése „megengedett”, amelyen a címkézés szoros, vagyis

$$v \in T_q, w \notin T_q, x_{v,w} = 0, \text{ és } d(v) = d(w) + 1,$$
$$v \notin T_q, w \in T_q, x_{v,w} = u_{v,w}, \text{ és } d(w) = d(v) + 1.$$

Egy ilyen belépő éllel (és a q kilépő éllel) elvégzett pivotálás az előfolyamos technológiával annak felel meg, hogy a feszítőfában kicseréljük a kilépő élt a belépő élre, majd a v aktív csúcsból s felé toljuk a lehető legtöbb folyamot az új $v \to s$ feszítőfabeli úton.

6. Algoritmus. Címkézéses duál szimplex (Armstrong, Chen, Goldfarb, Jin)

1. T egy tetszőleges feszítőfa $S = \{s\}$ részfával.
2. Az (S, Z) vágás élei legyenek duál fízibilisek, más bázison kívüli élre legyen $x_e = 0$.
3. Toljuk a többleteket s felé.
4. Legyen $d(s) = n$, $d(t) = 0$, $d(v) = 1$ minden $v \in V \backslash \{s, t\}$.
5. **amíg** létezik $g \in V$ aktív csúcs
6. Legyen g egy aktív csúcs, $q = pred(g)$.
7. **amíg** nincs p belépő élünk
8. Legyen $v \in T_q$ olyan, hogy $d(v) = \min_{w \in T_q} d(w)$.
9. **ha** létezik v-hez illeszkedő *megengedett* belépő él **akkor**
10. Legyen p egy v-hez illeszkedő megengedett él.
11. **egyébként**
12. Legyen $d(v) = d(v) + 1$.
13. **vége**
14. **vége**
15. $T = T \cup \{p\} \backslash \{q\}$, toljuk g többletét s felé.
16. **vége**

2.2. Tétel. (Armstrong, Chen, Goldfarb, Jin [6]) *A 6. algoritmus legfeljebb* $2nm$ *pivotálással helyesen megoldja a maximális folyam feladatot.*

Röviden összefoglaljuk a bizonyítás lépéseit.

Az első, bizonyítást érdemlő állítás szerint a csúcsok címkézése minden ciklus elején (az aktív csúcs kiválasztásakor) helyes. Ez kezdetben természetesen igaz, a fennmaradását pedig a legkisebb címkéjű csúcs vizsgálata garantálja.

Továbbá a duál fízibilitás is fennmarad az algoritmus során. Kezdetben ez is teljesül, később pedig akkor változhat, ha az (S, Z) vágásból származik a belépő

él. Ilyenre viszont csak akkor fanyalodunk, ha a Z-beli megengedett belépő élek elfogytak, hiszen a $Z \backslash T_q$-beli csúcsok címkéje kisebb, mint n, míg az S-beli csúcsoké legalább n marad. Így az aktív csúcsok elfogyásával optimális megoldást kapunk.

Végül a pivotálások számára kapott korlát a következő lemmán alapul:

2.2. Lemma. *Tegyük fel, hogy a 6. algoritmus során (u, v) egy pivotálás során belép a bázisba, legyen d a pivotálás előtti címkézés. Tegyük fel, hogy később kilép, majd még később ismét belép a bázisba, ezen pivotálás előtti címkézés legyen d'. Ekkor* $\min\{d'(u), d'(v)\} \geq \min\{d(u), d(v)\} + 1$.

Mivel a címkék monoton nőnek, és felülről korlátosak, így egy él legfeljebb $2n$-szer léphet be a bázisba, vagyis tényleg legfeljebb $2nm$ pivotálás történhet.

Az algoritmus implementálható $\mathcal{O}(n^2 m)$ futásidővel, illetve némi változtatással a műveletigény levihető a kombinatorikus előfolyam algoritmusok által gyakran elért $\mathcal{O}(n^3)$ futásidőre. A „pivotálások" száma ez utóbbi esetben is $\mathcal{O}(nm)$, viszont az aktív csúcsból nem toljuk el a többletet teljesen s-ig, így egy pivotálás $\mathcal{O}(1)$ időt vesz igénybe ([6], Algoritmus 3).

3. Nem nulla alsó korlátú feladatok

Ebben a fejezetben olyan maximális folyam feladatokat tekintünk, ahol a folyam alsó korlátja nem feltétlenül nulla, hanem egy l függvénnyel adott:

$$\begin{aligned} & \max x_{t,s} \\ \forall v \in V: \quad & \sum_{(w,v)\in E^*} x_{w,v} - \sum_{(v,w)\in E^*} x_{v,w} = 0 \\ \forall e \in E: \quad & l_e \leq x_e \leq u_e. \end{aligned} \tag{2}$$

A primál és duál hálózati szimplex algoritmusok működését a 2. fejezetben nemnulla alsó korlátokkal írtuk le.

Primál szimplex algoritmus esetén (2. és 5. algoritmus) az $x = 0$ folyammal és egy tetszőleges T feszítőfával indultunk el. Nem nulla alsó korlát esetén találnunk kell egy primál fízibilis bázismegoldást, ezek után mindkét algoritmus ugyanúgy működik tovább, a pivotálások számára adott korlát is változatlan marad a második algoritmusnál.

Duál szimplex algoritmus esetén (3. és 6. algoritmus) egy tetszőleges duál fízibilis bázismegoldással indulunk. Ilyet egyszerű konstruálni. Vegyünk egy tetszőleges T feszítőfát, állítsuk az (S, Z) vágásbeli éleket duál fízibilis folyamértékre, állítsuk a többi bázison kívüli él folyamértékét az alsó vagy felső korlátjára, majd számítsuk ki a bázisbeli élek folyamértékét. Így valóban duál fízibilis bázismegoldást kapunk, a bázisbeli élek primál fízibilitása persze nem biztosított. Ez a címkézés

nélküli, általános duál hálózati szimplex algoritmushoz megfelelő induló megoldás, azonban a címkézéses duál szimplex algoritmus [6] egy speciális duál megengedett megoldással indult, ami garantálta, hogy a folyam átírható előfolyam formára. Sajnos jelen esetben ez nem feltétlenül igaz.

Továbbá mivel nem nulla alsó korlátok esetén nem feltétlenül létezik megengedett megoldása a feladatnak, így a duál jellegű algoritmusokban valahol szerepelnie kell egy megállási feltételnek, ahol az algoritmus azt mondja, hogy a feladat nem megoldható, és leáll. Valóban, a duál szimplex algoritmusnál (3. algoritmus) lehetséges, hogy a kiválasztott kilépő élhez nem létezik lehetséges belépő él.

Legyen például a kilépő q élre $x_q > u_q$, és mutasson q az alatta levő T_q részfába. Ekkor lehetséges belépő él nemlétezése azt jelenti, hogy minden más T_q-ba befelé mutató élre $x_e = u_e$ (ezek bázison kívüli élek), illetve a T_q-ból kifelé mutató élekre $x_e = l_e$. T_q csúcsaiba (a megmaradási egyenleteket összegezve) ugyanannyi folyam folyik be, mint ki, tehát:

$$\sum_{(v,w):\overline{T}_q \to T_q} x_{v,w} = \sum_{(v,w):T_q \to \overline{T}_q} x_{v,w}.$$

Itt felhasználva a folyamértékekre vonatkozó információinkat (a bal oldalon $x_q > u_q$):

$$\sum_{(v,w):\overline{T}_q \to T_q} u_{v,w} < \sum_{(v,w):T_q \to \overline{T}_q} l_{v,w}.$$

Vagyis a T_q halmazból több folyamot kell kivinnünk, mint amennyit be lehet vinni. Tehát ha az algoritmus elakad a belépő él választásánál, akkor a feladat tényleg infízibilis. Sőt, erre kapunk egy egyszerűen ellenőrizhető bizonyítékot is a T_q halmazban. Ilyen halmaz létezése egyébként karakterizálja a hálózati folyam feladat megoldhatóságát:

3.1. Tétel. (Gale [21], Hoffman [30]) *A hálózati folyam feladatnak pontosan akkor létezik megengedett megoldása, ha minden $Q \subseteq V$ csúcshalmazra*

$$\sum_{(v,w):\overline{Q} \to Q} u_{v,w} \geq \sum_{(v,w):Q \to \overline{Q}} l_{v,w}.$$

3.1. Első fázis feladat

Visszatekintve a feladat felírására (2), természetes ötlet bevezetni az $x'_e = x_e - l_e$ változókat (és legyen $x'_{t,s} = x_{t,s}$), hiszen így az x' változókra nulla alsó korlátot kapunk. A csúcsokra vonatkozó egyenletek azonban megváltoznak:

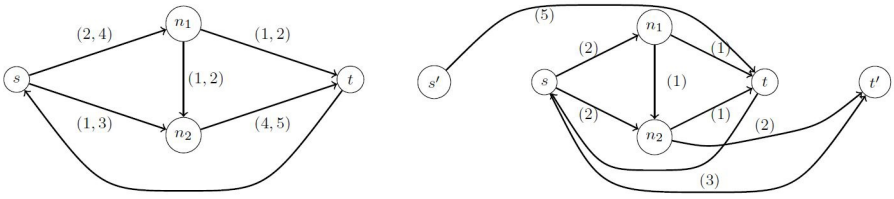
$$\forall v \in V : \sum_{(w,v)\in E^*} x'_{w,v} - \sum_{(v,w)\in E^*} x'_{v,w} = \sum_{(v,w)\in E^*} l_{v,w} - \sum_{(w,v)\in E^*} l_{w,v}. \qquad (3)$$

Itt (3) jobboldalát $b(v)$-vel jelölve a következő lineáris programozási feladatot kapjuk:

$$\begin{aligned} & & & \max x'_{t,s} \\ \forall v \in V: & & \sum_{(w,v)\in E^*} x'_{w,v} - & \sum_{(v,w)\in E^*} x'_{v,w} = b(v) \\ \forall e \in E: & & 0 \le x'_e \le u_e - l_e. & \end{aligned}$$

Itt $b(v)$ értelmezhető úgy, mint a csúcs igénye, illetve negatív érték esetén mint a csúcsban lévő $-b(v)$ többlet. Természetesen $\sum_{v\in V} b(v) = 0$ fennáll. Ismert módszer ilyen feladatok megengedett megoldásának előállítására a következő visszavezetés.

Vezessünk be két új csúcsot, jelölje őket s' és t'. Húzzuk be az (s', v) élt, ha $b(v) < 0$, és legyen ezen él felső korlátja $-b(v)$. Hasonlóan, húzzuk be a (v, t') élt, ha $b(v) > 0$, és legyen ezen él felső korlátja $b(v)$. Az új élek alsó korlátja mindkét esetben legyen 0. Tekintsük a 4. ábrán látható példát.



(a) az eredeti feladat (b) átalakítás után

4. ábra. Alsó korlátos feladat átalakítása.

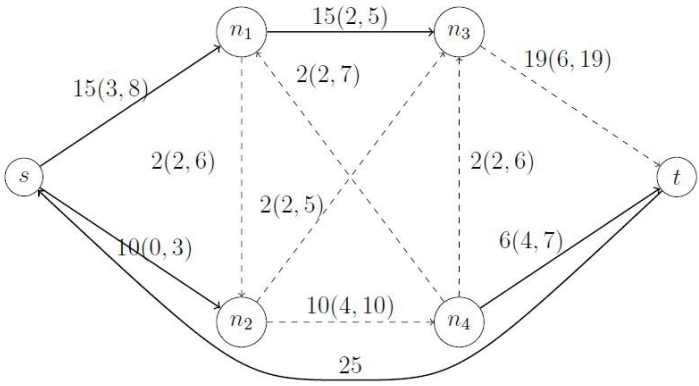
Könnyen látható, hogy 4a egy tetszőleges megoldása átalakítható 4b egy megoldásává az alsó korlátok levonásával, illetve az új élek folyamértékének a felső korláton való megválasztásával. Ekkor egyben 4b egy maximális $s' \to t'$ folyamát is kapjuk, hiszen például az s'-ből induló élek egy vágást alkotnak. Ennek megfelelően az eredeti feladatnak pontosan akkor van megengedett megoldása, ha az átalakított feladatot mint maximális folyam feladatot megoldva, a kapott folyam telíti az újonnan bevezetett éleket. Ebben az esetben az alsó korlátok visszaadásával és az új élek elvételével az eredeti feladat egy megengedett megoldását kapjuk.

Tehát megengedett megoldást előállíthatunk egy kicsit nagyobb ($n + 2$ csúcsú és legfeljebb $m + n$ élű, tehát nagyságrendileg nem nagyobb) maximális folyam feladat megoldásával. Amennyiben ezt pivotalgoritmussal végezzük, úgy a kapott megoldás is könnyen bázismegoldássá alakítható.

3.2. Fízibilitási MBU-algoritmus

Egy másik, lineáris programozásbeli megközelítéssel indulhatunk egy tetszőleges (nem feltétlenül primál vagy duál megengedett) bázisból, majd a primál nem megengedett éleket egyenként „kijavíthatjuk". Amennyiben a pivotalgoritmus során mindig egy kiválasztott nem megengedett élre koncentrálunk, és figyelünk arra, hogy primál megengedett élek ne váljanak nem megengedetté, akkor a fízibilitási MBU [8] egy specializációját kapjuk. Mivel a fízibilitási MBU során a valódi célfüggvénnyel nem foglalkozunk, felfoghatjuk úgy is az algoritmust, mintha az éppen kijavítani kívánt él folyamértékét maximalizálnánk ($x < l$ esetén, illetve az él megfordítását $x > u$ esetén) a nem megengedett éleket nem figyelembe véve a δ kiszámolásánál, és az algoritmust megállítjuk, amint élünk elérte a fízibilis intervallumot. Az élek kijavítását Goldfarb és Hao [26, 27] már ismertett címkézési technikájával (5. algoritmus) végezve erősen polinomiális algoritmust kapunk a feladatra [33].

Példaként tekintsük az 5. ábrát. Itt a zöld és kék élek alkotják a maximális folyam feladat bázisát. A jelenlegi bázismegoldás több primál nem megengedett élt is tartalmaz, ezek közül a kékkel jelölt (s, n_1) élt választottuk ki, őt javítjuk ki először.



5. ábra. MBU fízibilitási algoritmus.

Szeretnénk, ha a $(g, h) = (s, n_1)$ él kilépne a bázisból. Ekkor a feszítőfa két részre szakadna: $G = \{s, n_2, n_4, t\}$, illetve $H = \{n_1, n_3\}$. Mivel csökkenteni szeretnénk (s, n_1)-et, így olyan $G \to H$ él léphet be a bázisba, amelyen tudunk növelni (vagyis amelyre $x = l$), vagy olyan $H \to G$ él, amelyen tudunk csökkenteni (vagyis amelyre $x = u$). A megfelelő élek $\{(n_2, n_3), (n_3, t), (n_4, n_1), (n_4, n_3)\}$, közülük kell egy címkézés segítségével választanunk.

A primál szimplex algoritmus címkézéses változatához (5. algoritmus) hasonlóan szeretnénk címkézni. Jelen esetben az (s, n_1) élt szeretnénk csökkenteni, amit

az (n_1, s) „él” növeléseként is felfoghatunk, vagyis a (t, s) él szerepét az (n_1, s) él veszi át. Így tehát érdemes a címkézést az s csúcsból indítani, illetve jelen esetben az (s, n_1) él nem használható, ellenben a (t, s) él igen.

Általánosan, a (g, h) primál nem megengedett élhez történő címkézés megegyezik a 4. algoritmusban leírtakkal, annyi különbséggel, hogy a (t, s) él használható, de a (g, h) él nem, illetve a címkézés kiindulópontja $x_{g,h} > u_{g,h}$ esetén a g, míg $x_{g,h} < l_{g,h}$ esetén a h csúcs.

Az algoritmus vázlata a következő:

7. Algoritmus. Megengedett folyam keresése az MBU-algoritmussal [33]

1. T tetszőleges feszítőfa, x egy T-hez tartozó bázismegoldás.
2. **amíg** létezik nem megengedett él
3. Legyen (g, h) egy primál nem megengedett él.
4. **amíg** (g, h) nem megengedett
5. Címkézés (g, h)-hoz.
6. **ha** $d(h) = \infty$ $(d(g) = \infty)$ **akkor**
7. Nincs megengedett megoldás, megállunk.
8. **vége**
9. Belépő él: egy minimális címkéjű lehetséges belépő él.
10. Kilépő él: primál hányadosteszt a megengedett változókon.
11. Végezzük el a pivotálást.
12. **vége**
13. **vége**

Egy (g, h) él kijavításához legfeljebb nm pivotra van szükség (amennyiben nem akad el az algoritmus infízibilitás miatt), ennek bizonyítása nagyrészt megegyezik a címkézéses primál szimplex algoritmusnál [27] leírtakkal. Az algoritmus elején legfeljebb $n - 2$ primál nem megengedett változónk lehet (a feszítőfa $n - 1$ élből áll, és (t, s) nem lehet infízibilis). Mivel a kilépő élt mindig a megengedett éleken való primál hányadostesztte választjuk, így az algoritmus során nem válhat már megengedett él nem megengedetté. Ebből következően legfeljebb $n - 2$ javítási ciklus történik az algoritmusban, vagyis a pivotálások számára a következő korlátot kapjuk:

3.2. Tétel. (Illés, Molnár-Szipai [33]) *A 7. algoritmus $\mathcal{O}(n^2m)$ pivotálással megtalálja a maximális folyam feladat egy megengedett megoldását, vagy kimutatja, hogy a feladat nem megoldható.*

Megjegyezzük, hogy az algoritmus nem használja, hogy a (t, s) élt maximalizáljuk, tetszőleges célfüggvényű hálózati folyamhoz használható megengedett cirkuláció keresésére.

4. Primál MBU szimplex algoritmus

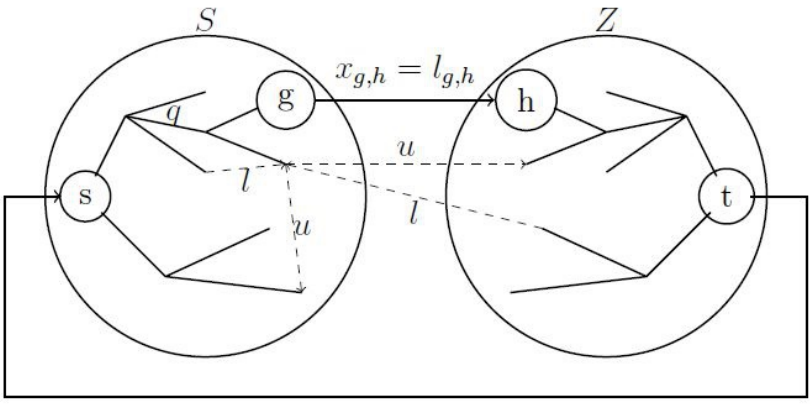
A primál MBU szimplex algoritmus [4] ötlete a primál szimplex algoritmushoz képest a következő. Egy tetszőleges duál nem megengedett változó kiválasztása után hányadosteszttel kiválasztjuk a kilépő változót, de nem végezzük el rögtön a pivotálást. Ekkor ugyanis duál megengedett változók duál infízibilissé válhatnak. Ezt elkerülendő, végzünk egy duál hányadosztesztet a kilépő változó során, és amennyiben kisebb duál hányadost kapunk, mint az eredetileg kiválasztott duál nem megengedett változón, akkor inkább ezt a pivotot lépjük meg.

8. Algoritmus. Primál MBU szimplex algoritmus LP-feladaton [4]

1. x primál megengedett bázismegoldás.
2. **amíg** létezik duál nem megengedett változó
3. Legyen x_{p^*} egy duál nem megengedett változó („vezérváltozó").
4. **amíg** x_{p^*} duál nem megengedett
5. Primál hányadosteszt p^* oszlopában a primál megengedett változókon:
6. $$q = \arg\min \left\{ \frac{\overline{b}_q}{\overline{a}_{q,p^*}} \ : \ \overline{a}_{q,p^*} > 0, \overline{b}_q \geq 0 \right\}.$$
7. Ha nem létezik q, melyre $\overline{a}_{q,p^*} > 0$ és $\overline{b}_q \geq 0$, akkor a feladat duál nem megengedett.
8. Legyen $\vartheta_1 = |\overline{c}_{p^*}|/\overline{a}_{q,p^*}$.
9. Végezzünk duál hányadosztesztet q sorában a duál megengedett változókon:
10. $$p = \arg\min \left\{ \frac{\overline{c}_p}{|\overline{a}_{q,p}|} \ : \ \overline{c}_p \geq 0, \overline{a}_{q,p} < 0 \right\}.$$
11. Legyen $\vartheta_2 = \overline{c}_p/|\overline{a}_{q,p}|$.
12. **ha** $\vartheta_2 < \vartheta_1$ **akkor**
13. Pivotáljunk $a_{p,q}$-n.
14. **egyébként**
15. Pivotáljunk $a_{p^*,q}$-n.
16. **vége**
17. **vége**
18. **vége**

Felmerülhet bennünk, hogy mivel a primál hányadostesztet a vezérváltozó oszlopán végeztük, de a $\vartheta_2 < \vartheta_1$ esetben nem ő lép be a bázisba, így a primál megengedettség nem feltétlenül marad fenn. Ez így is van, az algoritmus némileg meglepő tulajdonsága viszont az, hogy a vezérváltozó bázisba való belépése ($\vartheta_2 \geq \vartheta_1$) után, ami valamilyen ciklizálás elleni technika használata esetén véges sok pivotálás után megtörténik, a bázis ismét primál megengedetté válik. Továbbá a vezérváltozó bázisba lépésével a duál nem megengedett változók száma legalább 1-gyel csökkent, hiszen a duál hányadosteszt garantálja, hogy már duál megengedett változók nem válnak infízibilissé az algoritmus során. Így a duál megengedett változók halmaza „monoton módon növekszik", innen az algoritmus elnevezése (monotonic build up simplex algorithm).

Nézzük meg, hogyan változik az algoritmus, ha maximális folyam feladatra alkalmazzuk! A maximális folyam feladat primál megengedett bázismegoldásának struktúráját már ismerjük a primál szimplex algoritmusnál leírtakból. Ismerjük a duál nem megengedett él fogalmát, ilyen csak az (S, Z) vágásban lehet. A primál hányadosteszt a vezérváltozó belépésével kialakuló körben való javítás szűk keresztmetszetét választja ki. A 6. ábrán $x_{g,h}$ alsó korláton levő vezérváltozóval láthatunk egy példát.



6. ábra. Primál MBU hálózati szimplex

A duál hányadostesztnél történik lényeges egyszerűsödés a lineáris programozáshoz képest. A kilépő q élhez tartozó lehetséges p belépő élek duál hányadosa 0 (ha a p él S-en belül van), illetve 1 lehet (ha p az (S, Z) vágásban van). Mivel a vezérváltozó duál hányadosa is 1, így csak akkor nem engedjük a vezérváltozót belépni, ha létezik S-en belüli belépő él (Z-n belüli, ha $q \in Z$). Ha több ilyen lehetséges belépő él létezik, akkor címkézéssel fogunk köztük dönteni.

9. Algoritmus. Primál MBU-SA maximális folyam feladaton, címkézéssel [34]

```
1.  x primál megengedett bázismegoldás.
2.  amíg létezik duál nem megengedett él
3.      Legyen (g, h) egy duál nem megengedett él („vezérváltozó").
4.      amíg (g, h) duál nem megengedett
5.          Legyen q a primál hányadosteszttel kapott kilépő él.
6.          ha létezik q részfáján belüli lehetséges p belépő él akkor
7.              Válasszunk közülük minimális címkéjűt.
8.              Pivotáljunk: belép p, kilép q.
9.          egyébként
10.             Pivotáljunk: belép (g, h), kilép q.
11.         vége
12.     vége
13. vége
```

Vegyük észre, hogy egy (g, h) vezérváltozóhoz tartozó ciklus során végig S-en vagy Z-n belüli élek lépnek be (q helyzetétől függően), így az S és Z halmazok nem változnak, amíg a vezérváltozó be nem lép a bázisba, és a ciklus véget ér. Így tehát az S és Z részfát kezelhetjük külön egy cikluson belül, bevezethetünk külön címkézést a két részgráfon.

Tegyük fel, hogy $g \in S$ és $h \in Z$, ellenkező esetben g és h szerepe felcserélődik. Ha $v \in S$, akkor a $d(v)$ címke legyen a legrövidebb pszeudo-javítóút hossza v-ből g-be S-en belül, és ha $v \in Z$, akkor legyen a legrövidebb pszeudo-javítóút hossza h-ból v-be Z-n belül.

4.1. A primál MBU hálózati szimplex polinomialitása maximális folyam feladatra

Feladatunk egy vezérváltozó-ciklus lépésszámának megbecsülése, hiszen a duál megengedett élek számának monoton növése miatt legfeljebb annyi ciklus lesz, ahány duál nem megengedett él volt a kezdő bázismegoldásban ($\mathcal{O}(m)$).

A bizonyítás gondolatmenete azonos a korábban látott gondolatmenettel: megmutatjuk, hogy a címkék monoton növekednek, korlátosak, illetve egy él két bázisba való belépése között ténylegesen nőtt legalább az egyik csúcsának címkéje. Mivel a címkék egy h-tól, vagy g-be vezető legrövidebb pszeudo-javítóút hosszaként vannak definiálva, tényleg felülről becsülhetők n-nel, hiszen legrosszabb esetben is létezik egy, a feszítőfán belüli út, ami legfeljebb ilyen hosszú.

4.1. Lemma. (Monotonitás) *Tegyük fel, hogy a maximális folyam feladatot a 9. algoritmussal oldjuk meg. Egy vezérváltozó cikluson belül bármely $v \in V$ csúcsra és i iterációra igaz, hogy $d^{i+1}(v) \geq d^i(v)$.*

A lemma bizonyítása hasonló a 2.1. lemma bizonyításához (monotonitási lemma a Goldfarb–Hao-címkézéssel), részletei (az alfejezet többi állításának bizonyításával együtt) megtalálhatók a szerzők [34] cikkében. A címkék tényleges növekedéséről szóló lemma a következő alakú:

4.2. Lemma. (Fő lemma) *Tegyük fel, hogy a maximális folyam feladatot a 9. algoritmussal oldjuk meg. Ha egy vezérváltozó-cikluson belül (v, w) belépett az i. pivot során, kilépett a $i < j$. pivot során, majd ismét belépett a $j < k$. pivot során, akkor $d^{k+1}(v) + d^{k+1}(w) \geq d^i(v) + d^i(w) + 2$ teljesül (ahol $d^m(v)$ az m. pivotálás előtti címke).*

A lemma bizonyítása két részre bomlik aszerint, hogy a bázisba való két belépés során azonos irányban történt-e a belépés, vagyis ha például $(v, w) \in S$, és az i. pivot után v volt közelebb g-hez, akkor ugyanez igaz-e a k. pivot során történő belépés után is. Ezt a továbbiakban „felső csúcsnak" fogjuk nevezni, ami egybevág az elképzeléssel, hogy a két részfa a g, illetve h csúccsal van „fellógatva". Mindkét eset alapja a következő „részfa lemma" (4.3. lemma), de az egyik esetben szükség van a „megfordítási lemmára" (4.4. lemma) is.

4.3. Lemma. (Részfa lemma) *Tegyük fel, hogy a maximális folyam feladatot a 9. algoritmussal oldjuk meg. Ha (v, w) belépett a bázisba az i. pivot során w felső csúccsal, és ez így marad az $i \leq j$. pivot elvégzése után is, akkor minden $z \in T_v^{j+1}$ csúcsra $d^{j+1}(z) \geq d^i(w) + 1$.*

A lemma bizonyítása során teljes indukciót használunk, illetve kihasználjuk, hogy a lehetséges belépő élek közül mindig a minimális címkéjűt választjuk.

4.4. Lemma. (Megfordítási lemma) *Tegyük fel, hogy a maximális folyam feladatot a 9. algoritmussal oldjuk meg. Ha (v, w) belépett a bázisba az i. pivot során w felső csúccsal, és ez az $i < j$. pivot során megváltozik, akkor $d^{j+1}(w) \geq$ $\geq d^i(w) + 1$.*

A két segédlemmából összerakható a fő lemma bizonyítása, aminek birtokában már könnyen bizonyítható a következő felső korlát a pivotálások számára:

4.1. Tétel. *A címkézéses primál MBU szimplex algoritmus (9. algoritmus) legfeljebb $2nm^2$ pivotálással megoldja a maximális folyam feladatot.*

5. Duál MBU szimplex algoritmus

A duál MBU szimplex algoritmus [4] a primál MBU szimplex algoritmus duálisa. Röviden összefoglalva egy duál megengedett megoldásból indul, és egyenként kijavítja a primál nem megengedett változókat. Egy ilyen változó kijavítása során elveszhet a duál megengedettség, de a változó kilépésével helyreáll.

Az algoritmus pszeudo-kódja szintén nagyon hasonlít:

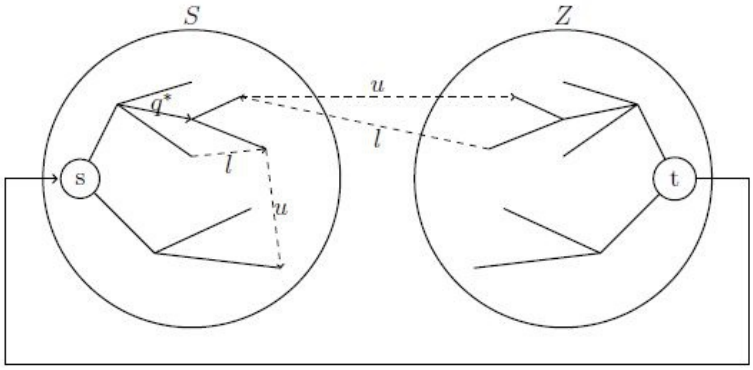
10. Algoritmus. Duál MBU szimplex algoritmus LP-feladaton [4]

1. x duál megengedett bázismegoldás.
2. **amíg** létezik primál nem megengedett változó
3. Legyen x_{q^*} egy infízibilis változó („vezérváltozó").
4. **amíg** x_{q^*} infízibilis
5. Duál hányadosteszt q^* sorában a duál megengedett változókon:
6. $$p = \arg\min \left\{ \frac{\overline{c}_p}{|\overline{a}_{q^*,p}|} \,:\, \overline{a}_{q^*,p} < 0, \overline{c}_p \geq 0 \right\}.$$
7. (Ha nem létezik p, melyre $\overline{a}_{q^*,p} < 0$ és $\overline{c}_p \geq 0$, akkor a feladat nem megoldható.)
8. Legyen $\vartheta_1 = \overline{b}_{q^*}/\overline{a}_{q^*,p}$.
9. Végezzünk primál hányadosteszt p oszlopában a primál megengedett változókon:
10. $$q = \arg\min \left\{ \frac{\overline{b}_q}{\overline{a}_{q,p}} \,:\, \overline{b}_q \geq 0, \overline{a}_{q,p} > 0 \right\}.$$
11. Legyen $\vartheta_2 = \overline{b}_q/\overline{a}_{q,p}$.
12. **ha** $\vartheta_2 < \vartheta_1$ **akkor**
13. Pivotáljunk $a_{p,q}$-n.
14. **egyébként**
15. Pivotáljunk a_{p,q^*}-on.
16. **vége**
17. **vége**
18. **vége**

A $\vartheta_2 < \vartheta_1$ esetben elromolhat a duál megengedettség, hiszen a duál hányadosztesztet q^* során végeztük, nem pedig q során, de a vezérváltozó kilépésével a duál megengedettség helyreáll. Így a duál MBU-algoritmus során a primál megengedett változók halmaza monoton növekszik, az utolsó ciklus után pedig optimális megoldást kapunk (ha fízibilis a feladat).

Ez két okból is szerencsés a mi helyzetünkben. Egyrészt kiinduló duál megengedett megoldást könnyen adhatunk nem nulla alsó korlátok esetén is. Másrészt a kijavítani kívánt primál infízibilis élek száma kezdetben $\mathcal{O}(n)$, hiszen ők csak a feszítőfában tartózkodhatnak, vagyis kevesebb ciklusra lesz szükség, mint a primál MBU-algoritmus esetében.

Mennyivel egyszerűsödik az algoritmus, ha maximális folyam feladatra alkalmazzuk? A vezérváltozónk egy infízibilis él a feszítőfában. Mint kilépő élt tekintve, a lehetséges belépő élek duál hányadosa 0 vagy 1 attól függően, hogy a vezérváltozó részfáján belül van, vagy az (S, Z) vágás egy telített éle. Tehát belépő élnek csak akkor választunk vágásbeli élt, ha más nincs, ettől eltekintve szabadon választhatunk. Végül a primál hányadosteszt eldönti, hogy a kialakuló körből kiléphet-e a vezérváltozónk a többi él megengedettségének megsértése nélkül. A 7. ábrán egy példát láthatunk a $q^* \in S$ és $x_{q^*} > u_{q^*}$ esetben.



7. ábra. Duál MBU hálózati szimplex

A belépő él kiválasztásában van némi szabadságunk, megfelelő címkézési technikával polinomiális futásidőt remélhetünk. A címkézés során figyelembe kell vennünk, hogy a duál hányadosteszt szerint a nem vágásbeli éleket kell preferálnunk. Ez első ránézésre jelentősen megnehezíti a dolgunkat. Tekintsük viszont az MBU-algoritmus korrektségére vonatkozó bizonyítást ([4], 1. tételének átírása a duál MBU-algoritmusra):

5.1. Tétel. *Tekintsük a duál MBU-algoritmus q^* vezérváltozó ciklusában történő $\vartheta_2 < \vartheta_1$ típusú pivotok egy sorozata után előálló megoldást. Ekkor a jelenlegi bázismegoldásra a következő három tulajdonság teljesülni fog:*

a) $\overline{b}_{q^*} < 0$.

b) Ha $\overline{c}_j < 0$, *akkor* $\overline{a}_{q^*,j} > 0$.

c) $\max_j \left\{ \frac{|\overline{c}_j|}{\overline{a}_{q^*,j}} : \overline{c}_j < 0 \right\} \leq \min_p \left\{ \frac{\overline{c}_p}{|\overline{a}_{q^*,p}|} : \overline{c}_p \geq 0, \overline{a}_{q^*,p} < 0 \right\}$.

Az a) tulajdonság szerint a vezérváltozó infízibilis marad, amíg a ciklus végén ki nem lép a bázisból (egy $\vartheta_1 \geq \vartheta_2$ típusú pivotálással).

A b) tulajdonság szerint a ciklus során duál infízibilissé vált változók akkor sem lehetnének lehetséges belépő élek, ha ezt egyébként nem kötöttük volna ki.

A c) tulajdonság szerint egy tetszőleges lehetséges belépő változó duál hányadosa legalább annyi, mint egy tetszőleges duál infízibilis változó duál hányadosának abszolút értéke. Vagyis ha elvégeznénk a pivotálást egy tetszőleges lehetséges belépő élen és a vezérváltozón, akkor helyreállna a duál fízibilitás (persze a primál hányadosteszt figyelembe vétele nélkül néhány változó fízibilitása elromolhatna).

Átírva a c) egyenlőtlenséget maximális folyam feladatra a következőt kapjuk:

$$\max_j\{1 : \overline{c}_j = -1\} \leq \min_p\{\overline{c}_p : \overline{c}_p \geq 0, \overline{a}_{q^*,p} = -1\}.$$

Itt a bal oldal $-\infty$, ha nincsen még duál infízibilis él a jelenlegi megoldásban. Ha viszont létezik legalább egy ilyen él, akkor a bal oldal 1, és így az egyenlőtlenség szerint minden lehetséges belépő él redukált költsége legalább 1, vagyis ekkor már csak vágásbeli élek vannak.

Végül figyeljük meg, hogy ha a belépő változó a vágásból származott, vagyis $\overline{c}_p = 1$, akkor a kilépő változó új redukált költsége $\overline{c}'_q = \overline{c}_q - \frac{\overline{c}_p}{\overline{a}_{q,p}} = 0 - \frac{1}{1} = -1$.

Összefoglalva, ha a vezérváltozó ciklusa során egyszer elfogynak a $\overline{c}_p = 0$ típusú belépő élek, akkor kiválasztunk egy $\overline{c}_p = 1$ típusú belépő élt, de ekkor a kilépő él duál infízibilis lesz, és így a következő pivotálás során sem lesz már $\overline{c}_p = 0$ belépő élünk. Tehát a vezérváltozó ciklusa két részre bomlik, egy $\overline{c}_p = 0$ és egy $\overline{c}_p = 1$ részre.

Az első részben lényegében csak q^* részgráfjában dolgozunk, így a szokásos címkézést megszoríthatjuk erre a részre. Amikor elfogynak a 0 hányadosú belépő élek, akkor elkezdhetjük az egész gráfot címkézni anélkül, hogy az ilyen élek újbóli felbukkanásától kéne tartanunk. A címkézés megegyezik a fízibilitási MBU-algoritmusnál (7. algoritmus) leírtakkal. Így a következő algoritmust kapjuk:

11. Algoritmus. Duál MBU-SA maximális folyam feladaton címkézéssel [40]

1. x duál megengedett bázismegoldás.
2. **amíg** x nem primál megengedett
3. Legyen q^* egy primál nem megengedett él („vezérváltozó”).
4. Fízibilitási MBU-címkézéssel q^* részgráfján
5. **ha** q^* infízibilis **akkor**
6. Fízibilitási MBU-címkézéssel az egész gráfon
7. **vége**
8. **ha** q^* infízibilis **akkor**
9. A feladat infízibilis, megállunk
10. **vége**
11. **vége**
12. A jelenlegi megoldás optimális, megállunk

A fízibilitási MBU-algoritmusra belátott lépésszám korlátokból könnyen összerakható a 11. algoritmus lépésszámbecslése:

5.2. Tétel. (Molnár-Szipai [40]) *A címkézéses duál MBU-algoritmus legfeljebb $2n^2m$ pivotálással megoldja a maximális folyam feladatot.*

6. Mozdony hozzárendelési probléma

A MÁV-TRAKCIÓ Zrt. (2014 óta a MÁV-START része) és a BME Optimalizálási Csoportja egy kutatás-fejlesztési projekt keretében vizsgálta a következőkben ismertetett mozdony hozzárendelési problémát [35]. A vállalat vasúti vontatási feladatok ellátásával foglalkozik. A személyszállítás területén a megrendelő a MÁV-START, míg teherszállítás esetén számos cég foglalkozik vasúti fuvarozással. A MÁV-TRAKCIÓ Zrt. megállapodik a megrendelővel, hogy az ő vasúti kocsikra pakolt szállítmányát egy adott helyen és időben felveszi, majd a saját mozdonyai segítségével elszállítja egy másik adott helyre és időre.

A teherszállítást a személyszállítással szemben az jellemzi, hogy a megrendelések többféle bizonytalanságot mutatnak: (i) gyakoriak a késések, (ii) előfordulnak lemondások, (iii) időnként a szállítás időpontjához nagyon közel adják le a megrendelést. Mindezek negatívan befolyásolják az optimális tehervontatás megszervezését. A teherszállítás esetén, mivel nem menetrend alapú a közlekedés megszervezése, szükségszerű az ún. *gépmenetek* nagyobb számának a használata. A tehervonatok közlekedtetése esetén kétféle logikus cél merülhet fel: 1. minél kevesebb mozdony segítségével végezzük el a feladatot, 2. minél kevesebb gépmenet kilométert fussanak a mozdonyaink. Matematikai szempontból az első esetben maximális folyam feladatot kapunk, míg a második esetben minimális költségű folyam feladatot. Ezeket a feladatokat megoldó, specializált algoritmusok jelentősen eltérnek egymástól. Mivel ebben a tanulmányban nem célunk a vasút optimalizálási feladat részletekbe menő tárgyalása és megoldása, így mi csak azzal az esettel foglalkozunk, amikor a cél minél kevesebb mozdony segítségével elvégezni a vontatást. Tehát a vasút optimalizálási problémát csak a maximális folyam feladat polinomiális pivot algoritmusai hatékonyságának illusztrálására használjuk fel. Az optimalizálási feladatot tehát egy maximális (illetve minimális) folyam feladatként fogalmazzuk meg, és arra az egyszerűbb kérdésre keressük a választ, hogy legalább hány mozdonyra van szükség az elvállalt vontatások elvégzéséhez.

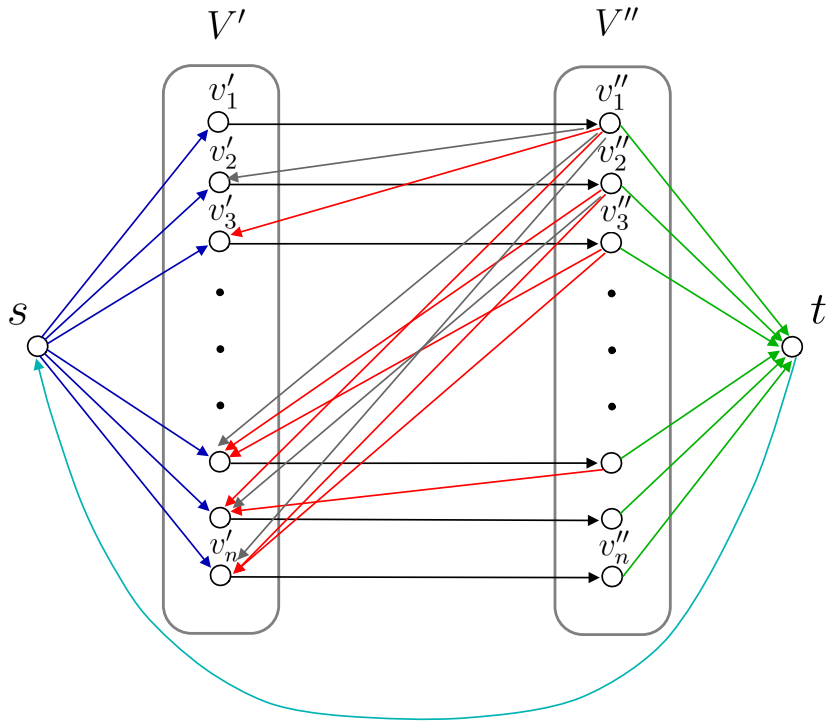
A matematikai modell

Legyen a teljesítendő megbízások halmaza V. Minden $v \in V$ megbízáshoz rendelkezésünkre állnak a következő adatok: indulási hely, indulási idő, érkezési hely, érkezési idő, használható mozdonytípusok. A felépített matematikai modell egy hálózati folyam, melyben minden v megbízásnak megfelel egy v' és v'' csúcs,

illetve egy (v', v'') él. Ha egy mozdony a v megbízás teljesítése után képes elvégezni a w megbízást, akkor felveszünk egy (v'', w') élt. Ez akkor teljesül, ha a v elvégzése után van elég idő az esetleges gépmenetre v érkezési helyéről w indulási helyére (d), illetve a szerelvények összerakására, ellenőrzésre (τ „technikai idő"):

$$t_v^{\text{érk}} + d(p_v^{\text{érk}}, p_w^{\text{ind}}) + \tau \leq t_w^{\text{ind}}.$$

Bevezetünk továbbá egy s forrást és t nyelőt, és felveszünk minden v-re egy (s, v') és (v'', t) élt. Végül bevezetünk egy (t, s) élt. Az így kapott hálózati folyamot szemlélteti a 8. ábra.



8. ábra. Minimális költségű hálózati folyam modell

Látható, hogy egy 1 értékű s-ből t-be menő folyam megfeleltethető egy úgynevezett mozdonyfordulónak, vagyis egy mozdony által elvégzett feladatsorozatnak. Megkövetelve, hogy mindegyik (v', v'') típusú élen legalább 1 legyen a folyamérték, a modell megengedett egészértékű megoldásai leírják a valós feladat megoldásait.

A modellről részletesebben [32]-ben olvashatunk.

Implementáció és eredmények

Jellemző méretű feladat a havi terv. Mivel a személyzetnek 15 nappal a hónap kezdete előtt meg kell adni egy beosztástervet, így a következő hónap tervét a jelenlegi hónap közepén rendelkezésre álló adatok alapján kell optimalizálni. Ez persze szükségessé tesz későbbi változtatásokat, de a módszer hatékonyságát jól méri.

A teszteléshez három mozdonytípus (V43, M62, M63) három havi (szeptember, október, november) megrendeléseit használtuk. Induló bázismegoldásként azt a megoldást használtuk, ahol mindegyik feladatot egy külön mozdony vontat el. Megjegyezzük, hogy csak a legfeljebb 3 nappal későbbi feladatokat kötöttük össze a modellezésnél leírtak szerint; evvel a megoldás minősége várhatóan nem romlik, viszont a gráf mérete jelentősen csökken. Kétféle induló bázismegoldást használtunk. Az első induló megoldásban minden feladatot külön mozdony vontat, míg a második megoldásban egy egyszerű mohó heurisztikával javítottunk ezen (ez a legnagyobb modell esetén is kevesebb, mint 6 másodperc alatt lefutott). Az így kapott modellek dimenzióit a 9. ábra tartalmazza.

Mozdonytípus	feladatok száma	csúcsok száma	élek száma	2. induló megoldás
szeptember V43	2850	5702	726338	93 mozdony
szeptember M62	1450	2902	175145	65 mozdony
szeptember M63	1789	3580	292260	63 mozdony
október V43	2901	5804	738650	94 mozdony
október M62	1400	2802	163437	59 mozdony
október M63	1780	3562	285932	59 mozdony
november V43	2816	5634	716115	90 mozdony
november M62	1403	2808	164792	62 mozdony
november M63	1742	3486	278817	60 mozdony

9. ábra. Használt adathalmazok.

Az algoritmusokat C#-ban implementáltuk. Teszteltük a szimplex algoritmust minimál index és Goldfarb–Hao-címkézés szerinti belépő él választással. Teszteltük az MBU-algoritmust minimál index, illetve Goldfarb–Hao-címkézéssel választott vezérváltozóval, kombinálva a belépő él minimál index, illetve címkézés szerinti választásával. Az eredmények, mind a pivotálások száma, mind a futásidő (perc:másodperc, illetve pivotálások száma) a 10. ábrán találhatóak az első induló megoldással, a 11. ábrán a második induló megoldással. A tesztelés egy Intel Core i7-3630QM processzoron történt.

	szimplex min	*szimplex GH*	*MBU min/min*	*MBU GH/min*	*MBU min/cmke*	*MBU GH/cmke*
szept. 43	42:43	**12:47**	30:15	32:58	36:56	40:08
	327720	352432	**62771**	97677	361617	330575
szept. 62	2:40	**1:10**	4:18	2:21	2:38	2:36
	80969	86627	45476	**31666**	83518	81258
szept. 63	6:49	**2:33**	8:02	5:36	7:09	6:54
	127084	143782	52859	**40505**	129703	133742
okt. 43	44:01	**12:30**	30:16	35:47	47:09	40:05
	329907	357003	**64011**	104258	365251	336215
okt. 62	2:20	**1:03**	3:14	1:50	2:21	2:28
	76010	81628	36450	**25149**	79154	77389
okt. 63	6:39	**2:34**	10:09	5:38	7:02	6:19
	125990	142162	62604	**41392**	146698	131238
nov. 43	43:58	**12:06**	26:54	30:56	48:24	38:22
	325876	347392	**55468**	92563	351603	327108
nov. 62	2:21	**1:03**	3:09	1:49	2:38	2:24
	76050	81848	37760	**28858**	78040	77873
nov. 63	6:10	**2:24**	11:22	6:00	6:24	5:57
	119792	138085	72000	**39418**	136417	126316

10. ábra. Eredmények primitív induló megoldással.

A mozdony hozzárendelési feladatból egyszerű struktúrájú, de nagyon degenerált feladatokat kapunk. Az iterációk jelentős száma, mondhatnánk túlnyomó többsége, primál degenerált iteráció.

Egy oszlopon belül jól megfigyelhető a feladat méretének hatása mind a futásidőre, mind a pivotálások számára. A legtöbb oszlop az élek számában lineáris pivotszámot mutat (kivétel talán az első táblázat harmadik oszlopa). Például a második induló megoldással, minimál indexes szimplex algoritmussal (időben leggyorsabb algoritmus) kapott pivotszámokra négyzetes értelemben leginkább illeszkedő $a + b \cdot m^c$ görbe a $6711 + 0{,}007 \cdot m^{1{,}1}$, míg a Goldfarb–Hao-szimplexre (legkevesebb pivotálás) a $2434 + 0{,}011 \cdot m^{0{,}98}$.

A heurisztikus induló megoldás jelentősen csökkentette az elvégzett pivotálások számát. Míg az első induló megoldás esetén a minimál indexes MBU-algoritmusoknak (3. és 4. algoritmus) jelentősen kevesebb iterációra volt szükségük, míg a második induló bázisnál a Goldfarb–Hao-szimplex pivotszáma tűnik csak ki a többi algoritmus közül.

	szimplex min	*szimplex GH*	*MBU min/min*	*MBU GH/min*	*MBU min/cmke*	*MBU GH/cmke*
szept. 43	**2:45**	3:48	14:52	16:11	26:17	21:34
	27630	**8721**	28291	28073	21276	22281
szept. 62	**0:20**	0:21	1:14	1:16	2:04	1:51
	11223	**4113**	11271	10921	9672	10126
szept. 63	**0:41**	0:43	2:46	2:51	4:33	4:24
	14708	**4904**	14967	14716	11633	12553
okt. 43	**2:44**	3:51	14:32	15:13	23:58	24:07
	28178	**8798**	28747	28549	21649	23491
okt. 62	**0:17**	**0:17**	1:04	1:05	1:22	1:37
	10369	**3765**	10595	10390	8326	9315
okt. 63	**0:36**	0:40	2:20	2:28	3:28	3:39
	13490	**4820**	13700	13213	10768	11042
nov. 43	**2:14**	3:41	14:13	14:16	21:33	21:26
	26280	**8695**	26948	26976	20561	21960
nov. 62	**0:19**	**0:19**	1:15	1:12	1:29	1:29
	10970	**4005**	11266	10846	8655	9117
nov. 63	**0:39**	0:42	2:34	2:36	3:39	3:44
	14028	**5002**	14118	13909	11245	11859

11. ábra. Eredmények heurisztikus induló megoldással.

Összehasonlítva a szimplex- és az MBU szimplex algoritmusok iterációs számait és futási idejeit, két megállapítást tehetünk: 1. Mivel az MBU szimplex algoritmus két hányadostesztet végez, és bonyolultabb, így nem meglepő, hogy jelentősen elmaradnak a futási idejei a szimplex variánsokétól. 2. Mivel az MBU szimplex algoritmusnál a duál megengedetté vált változók azok is maradnak, így a triviális bázisról indítva, sokszor van olyan variánsuk, amelyik sokkal kevesebb iterációval oldja meg a feladatot. Ezt az előnyüket a heurisztikus, közel optimális bázisról indítva már elveszítik.

A szimplex algoritmusvariánsokat összehasonlítva elmondható, hogy a triviális bázisról indulva a Goldfarb–Hao-féle címkézésnek jelentős hatása van abban, hogy jobb futási idők legyenek, közel azonos iterációszám mellett. A második, heurisztikus bázis esetén ez az előny eltűnik, és általában a minimál indexes szimplex algoritmus produkálja a leggyorsabb futásokat, annak ellenére, hogy a Goldfarb–Hao-féle címkézést használó szimplex algoritmus iterációszáma, a másik változat iterációs számának 25–35%-át produkálja csupán.

Érdekességképpen megjegyezzük, hogy a heurisztikus bázisról indítva az algoritmusokat az MBU szimplex variánsok iterációszámban mutatkozó sikeressége eltűnik, hiszen az összes algoritmusvariáns közel azonos iterációszámot produkál, kivéve a Goldfarb–Hao-féle címkézést használó szimplex algoritmust, amelyik iterációszámok tekintetében minden más algoritmusnál jobban teljesít.

Mindkét bázis esetén megállapítható, hogy az itt vizsgált feladatok esetén a szükséges lépésszám, az iterációk száma az összes algoritmusváltozat esetén jelentősen kisebb, mint az erősen polinomiális algoritmusváltozatok elméleti korlátja. Érdekességképpen jegyeznénk meg, hogy az MBU-szimplexvariánsok között az első bázison a legkisebb iterációszámot egyetlen egyszer sem produkálta az elméletileg erősen polinomiális variáns (a 6. algoritmus). Sőt a második bázis esetén sem volt olyan feladat, amelyik esetén az elméletileg erősen polinomiális MBU-szimplexváltozat (6. algoritmus) produkálta volna a legkisebb lépésszámot.

A szimplexvariánsok között már érdekesebb a verseny. A Goldfarb–Hao-féle címkézést használó szimplex algoritmus erősen polinomialitása ismert [27]. Annak ellenére, hogy az első bázis esetén, a feladatok megoldásához szükséges iterációk száma mindig magasabb volt, mint a minimál indexes szimplex algoritmusnál, a futási idők mindig kisebbek voltak. Ezzel szemben a heurisztikus bázisról indulva a minimál indexes szimplex algoritmus bizonyult gyorsabbnak, de a Goldfarb–Hao-féle címkézést használó szimplex algoritmus iteráció száma jelentősen kisebb volt.

A futásidők elemzésénél óvatosságra intjük az olvasót, ugyanis a futásidőt erősen befolyásolja az implementáció minősége (más tényezők mellett), hiszen számos olyan területe van a szimplex- és MBU szimplex algoritmusnak, ahol az ügyes implementációnak (pl. címkézés, minimál index kiválasztása egy halmazból) komoly hatása lehet a futásidőre. Általánosságban talán annyi mégis elmondható, hogy a címkézést használó algoritmusok a korrekt pivot pozíció megtalálásához több számítást végeznek, ami a legügyesebb implementációk esetén is növelheti a futásidőt. Ez főleg a második induló megoldással kapott eredményeken figyelhető meg.

7. Összefoglalás és további kutatási irányok

A dolgozat első felében összefoglaltuk a maximális folyam feladatra adott polinomiális primál és duál szimplex algoritmusok [6, 27] előzményeit és technikáit. Megmutattuk, hogy a gyakorlatban gyakran előforduló nem nulla alsó korlátok esetében mit lehet tenni. Az egyik megoldás egy segédfeladat megoldása megengedett megoldás keresésére, míg a másik a fízibilitási MBU-algoritmus, melyről megmutattuk, hogy szintén van polinomiális változata. Ezután beláttuk, hogy a primál és duál MBU szimplex algoritmusok is polinomiálissá tehetők a megfelelő címkézési technika alkalmazásával.

A primál és duál szimplex algoritmusokkal ellentétben az MBU szimplex algoritmusok áthaladnak se nem primál, se nem duál fízibilis bázismegoldásokon. A duál MBU-algoritmus további előnye, hogy alsó korlátos feladat esetén is első fázis feladat nélkül elindítható.

Érdekes lenne további lineáris programozási pivot algoritmusok [50] polinomialitását megvizsgálni maximális folyam feladatra. Az első, természetes jelölt ilyen vizsgálatra a criss-cross algoritmus [49] lenne, amelyiknél a gond a nem strukturáltan előforduló se nem primál, se nem duál bázisok jelentkezése. További általánosítási irány a feladatosztály bővítése lehetne. Minimális költségű hálózati folyam feladat esetén elveszik a redukált költségek egyszerű struktúrája, vajon így is polinomiálissá tehetők a lineáris programozás területéről ismert pivot algoritmusok? A primál és duál szimplex algoritmusoknak létezik ilyen variánsa [42, 43], bár jóval bonyolultabb technikával igazolható az erősen polinomialitás, mint a maximális folyamnál látottak.

Hivatkozások

[1] R. K. Ahuja, M. Kodialam, A. K. Mishra, and J. B. Orlin: *Computational investigations of maximum flow algorithms*, European Journal of Operational Research **97** (1997), 509–542.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin: *Network Flows*, Prentice Hall, New Jersey, 1993.

[3] R. J. Anderson and J. C. Setubal: *Network Flows and Matching: First DIMACS Implementation Challenge, chapter Parallel and sequential implementations of maximum-flow algorithms*, American Mathematical Society, 1993.

[4] K. M. Anstreicher and T. Terlaky: *A monotonic build-up simplex algorithm for linear programming*, Operations Research **42** (1994), 556–561.

[5] V. Árgilán, J. Balogh, J. Békési, B. Dávid, G. Galambos, M. Krész, and A. Tóth: *Ütemezési feladatok az autóbuszos közösségi közlekedés operatív tervezésében: Egy áttekintés*, Alkalmazott Matematikai Lapok **31** (2014), 1–40.

[6] R. D. Armstrong, W. Chen, D. Goldfarb, and Z. Jin: *Strongly polynomial dual simplex methods for the maximum flow problem*, Mathematical Programming **80** (1998), 17–33.

[7] Zs. Barta: *Vasút optimalizálási problémák: matematikai módszerek és modellek*, Diplomamunka, Budapesti Muszaki és Gazdaságtudományi Egyetem, 2011.

[8] F. Bilen, Zs. Csizmadia, and T. Illés: *Anstreicher–Terlaky type monotonic simplex algorithms for linear feasibility problems*, Optimisation Methods and Software **22(4)** (2007), 679–695.

[9] R. G. Bland: *New finite pivoting rule for the simplex method*, Mathematics of Operations Research **29(6)** (1981), 1039–1091.

[10] A. Charnes: *Optimality and degeneracy in linear programming*, Econometrica **20** (1952), 160–170.

[11] Zs. Csizmadia, T. Illés, and A. Nagy: *The s-monotone index selection rules for pivot algorithms of linear programming*, European Journal of Operational Research **221(3)** (2012), 491–500.

[12] W. H. Cunningham: *Theoretical properties of the network simplex method*, Mathematics of Operations Research **4(2)** (1979), 196–203.

[13] G. B. Dantzig: *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

[14] U. Derigs and W. Meier: *Implementing Goldberg's max-flow algorithm: A computational investigation*, Zeitschrift für Operations Research **33** (1989), 383–403.

[15] E. A. Dinic: *Algorithm for solution of a problem of maximum flow in networks with power estimation*, Soviet Math. Doklady **11** (1970), 1277–1280.

[16] J. Edmonds and R. M. Karp: *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of ACM **19** (1972), 248–264.

[17] L. R. Ford and D. R. Fulkerson: *A simple algorithm for finding maximal network flows and an application to the Hitchcock problem*, Technical report, Research Memorandum RM-1604, The RAND Corporation, Santa Monica, California, 1955.

[18] L. R. Ford and D. R. Fulkerson: *Maximal flow through a network*. Canadian Journal of Mathematics **8** (1956), 399–404.

[19] L. R. Ford and D. R. Fulkerson: *Flows in Networks*, Princeton University Press, Princeton, NJ., 1962.

[20] A. Frank: *Connections in Combinatorial Optimization*, volume **38** of Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2011.

[21] *D. Gale*: *A theorem on flows in networks*, Pacific Journal of Mathematics **7(2)** (1957), 1073–1082.

[22] *Z. Galil and A. Namaad*: *An* $\mathcal{O}\left(nm\log^2 n\right)$ *algorithm for the maximum flow problem*, Journal of Computer and System Sciences **21** (1980), 203–217.

[23] A. V. Goldberg, M. D. Grigoriadis, and R. E. Tarjan: *Use of dynamic trees in a network simplex algorithm*, Mathematical Programming **50** (1991), 277–290.

[24] A. V. Goldberg and R. E. Tarjan: *A new approach to the maximum flow problem*, In Proceedings of the 18th ACM Symposium on the Theory of Computing, pages 136–146, 1986.

[25] A. V. Goldberg and R. E. Tarjan: *A new approach to the maximum flow problem*, Journal of the ACM **35** (1988), 921–940.

[26] D. Goldfarb and J. Hao: *A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and* $\mathcal{O}(n^2m)$ *time*, Mathematical Programming **47** (1990), 353–365.

[27] D. Goldfarb and J. Hao: *On strongly polynomial variants of the network simplex algorithm for the maximum flow problem*, Operations Research Letters **10** (1991), 383–387.

[28] T. E. Harris and F. S. Ross: *Fundamentals of a method for evaluating rail net capacities*, Technical report, Research Memorandum RM-1573, The RAND Corporation, Santa Monica, California, 1955.

[29] *D. S. Hochbaum*: *The pseudoflow algorithm: A new algorithm for the maximum-flow problem*, Operations Research **56(4)** (2008), 992–1009.

[30] A. Hoffman: *Some recent applications of the theory of linear inequalities to extremal combinatorial analysis*, Proceedings Symposium Applied Mathematics **10** (1960), 113–128.

[31] T. Illés: *Lineáris optimalizálás elmélete és pivot algoritmusai*, Technical report, ELTE Operations Research Report, 2013-03.

[32] T. Illés, M. Makai, and Zs. Vaik: *Combinatorial optimization model for railway engine assignment problem*, In L. G. Kroon and R. H. Möhring, editors, Proceedings of the 5th Workshop on Algorithmic Methods and Models for Optimization of Railways, 2006.

[33] T. Illés and R. Molnár-Szipai: *On strongly polynomial variants of the MBU-simplex algorithm for a maximum flow problem with non-zero lower bounds*, Optimization **63** (2014), 39–47.

[34] T. Illés and R. Molnár-Szipai: *Strongly polynomial primal monotonic build-up simplex algorithm for maximal flow problems*, Discrete Applied Mathematics **214** (2016), 201–210.

[35] T. Illés and R. Molnár-Szipai: *Mozdonyhozzárendelés a vasúti teherszállításban*, Érintő Elektronikus Matematikai Lapok, 2017 június.

[36] A. V. Karzanov: *Nakhozhdenie maksimal'nogo potoka v seti metodom predpotokov („determining the maximal flow in a network by the method of preflows")*, Doklady Akademii Nauk SSSR **215(1)** (1974), 49–52.

[37] E. Klafszky: *Hálózati folyamok*, Bolyai János Matematikai Társulat, 1969.

[38] E. Lawler: *Kombinatorikus optimalizálás: hálózatok és matroidok*, Műszaki kiadó, 1982.

[39] I. Maros: *A practical anti-degeneracy row selection technique in network linear programming*, Annals of Operations Research **47** (1993), 431–442.

[40] R. Molnár-Szipai: *Dual MBU simplex algorithm for maximum flow problems (elfogadva)*, Acta Univ. Sapientiae Mathematica, 2017.

[41] J. B. Orlin: *Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem*, Technical report, Technical Report 1615-84, Sloan School of Management, MIT, Cambridge, MA., 1984.

[42] J. B. Orlin: *A polynomial time primal network simplex algorithm for minimum cost flows*, Mathematical Programming **78** (1997), 109–129.

[43] J. B. Orlin, S. Plotkin, and E. Tardos: *Polynomial dual network simplex algorithms*, Mathematical Programming **60** (1993), 255–276.

[44] F. Piu and M. G. Speranza: *The locomotive assignment problem: a survey on optimization models*, International Transactions in Operational Research **21(3)** (2014), 327–352.

[45] A. Prékopa: *Lineáris programozás 1*, Bolyai János Matematikai Társulat, 1968.

[46] A. Schrijver: *On the history of the transportation and maximum flows problems*, Mathematical Programming **5** (2002), 126–131.

[47] Y. Shiloach: *An $\mathcal{O}(nI \log^2 I)$ maximum flow algorithm*, Technical report, STAN-CS-78-702, Computer Science Department, Stanford University, Stanford, CA, 1978.

[48] D. D. Sleator and R. E. Tarjan: *Adata structure for dynamic trees*, Journal of Computer and System Sciences **26** (1983), 362–391.

[49] T. Terlaky: *A convergent criss-cross method*, Math. Oper. und Stat. ser. Optimization **16** (1985), 683–690.

[50] T. Terlaky and S. Zhang: *Pivot rules for linear programming: a survey on recent theoretical developments*, Annals of Operations Research **46(1)** (1993), 203–233.

[51] A. N. Tolstoi: *Planirovanie Perevozok, Sbornik pervyi*, chapter Metody nakhoždeniya naimen'šego summovogo kilometraža pri planirovanii perevozok v prostranstve, pages 23–55. Transpechat' NKPS, 1930.

[52] U. Zwick: *The smallest networks on which the Ford–Fulkerson maximum flow procedure may fail to terminate*, Theoretical Computer Science **148** (1995), 165–170.

Illés Tibor az ELTE TTK-n kapta matematikus diplomáját 1987-ben, egyetemi doktori címét 1989-ben, majd Phd fokozatát 1996-ban. Először az MTA-SZTAKI kutatója, majd 1990-től 2016-ig az ELTE Operációkutatás Tanszék oktatója. 2010 óta a BME TTK egyetemi docense, 2011-től a BME Differenciálegyenletek Tanszék vezetője. Kutatási területei a lineáris és nemlineáris programozás és ezek ipari és gazdasági alkalmazásai. Több, mint 60 közleményére 500-nál is több hivatkozást kapott, h-indexe 12. Témavezetésével, közel 60 hallgató készítette el szakdolgozatát 2 magyar és 2 külföldi egyetemen. Négy hallgatója szerzett doktori fokozatot.

Díjai és ösztöndíjai: Farkas Gyula-emlékdíj (1991), DAAD-ösztöndíj (1998), Bolyai Farkas-ösztöndíj (2001), Bolyai János kutatási ösztöndíj (2000–2003). Több alapkutatási és fejlesztési projektnek volt résztvevője vagy vezetője Magyarországon és külföldön, beleértve piacvezető hazai és külföldi nagyvállalatok számára végzett projekteket is.

Alapító tagja a Magyar Operációkutatási Társaságnak (1991) és az európai folytonos optimalizálók munkacsoportjának (EUROPT WG, 2000). Az EUROPT WG tiszteletbeli koordinátora (2003) és a MOT elnöke (2011–2014) és alelnöke (2014–2017). Tagja a BJMT-nek és a BJMT Alkalmazott Matematika Szakosztályának, titkára 1991 és 1993 között. Az EURO Végrehajtó Bizottságának tagja 2011-től. Az MTA Operációkutatási Tudományos Bizottságának 2005 óta tagja,

2008–2010 között a titkára. Az MTA Közgyűlésének választott képviselője 2013 óta.

ILLÉS TIBOR
Budapesti Műszaki és Gazdaságtudományi Egyetem
illes@math.bme.hu

Molnár-Szipai Richárd 1988-ban született. Alkalmazott matematikusként végzett a Budapesti Műszaki és Gazdaságtudományi Egyetemen. Kutatási területe a hálózati folyam modellek elmélete, és kombinatorikus algoritmusainak kapcsolata a lineáris programozás algoritmusaival. Jelenleg szoftverfejlesztő mérnökként dolgozik a Mentor Graphicsnál.

MOLNÁR-SZIPAI RICHÁRD
Budapesti Műszaki és Gazdaságtudományi Egyetem
mricsi@math.bme.hu

STRONGLY POLYNOMIAL PIVOT ALGORITHMS FOR MAXIMAL FLOW PROBLEMS

Tibor Illés and Richárd Molnár-Szipai

In this article we describe labelling techniques applied to the maximum flow problem. These can be traced back to the shortest augmenting path algorithm, and later used to prove the polynomiality of various pivot algorithms. We discuss the primal and dual simplex algorithms, as well as variants of the MBU algorithm in a unified system, with an added emphasis on handling nonzero lower bounds that arise frequently in applications. We compare the algorithms on railway engine assignment problems.