Eötvös Loránd Tudományegyetem

Természettudományi Kar

Kránicz Enikő Gréta

Lineáris kombinatorikus törtfüggvény optimalizálási feladatok

BSc Szakdolgozat

Témavezető:

Jüttner Alpár

Operációkutatási Tanszék

Budapest, 2013

Köszönetnyilvánítás

Ezúton szeretném megköszönni Jüttner Alpárnak, a konzulensemnek, hogy érdeklődésével felkeltette figyelmemet a téma iránt, valamint az útmutatásokat, és hasznos tanácsokat.
Köszönettel tartozom ezen kívül a családomnak, páromnak és barátaimnak türelmükért, bíztató szavaikért és tanácsaikért.

Budapest, 2013 május

Kránicz Enikő Gréta

Tartalomjegyzék

1. fejezet

Bevezetés

A kombinatorikus optimalizálás az alkalmazott matematika egy lendületesen fejlődő területe, amely a kombinatorika, lineáris programozás és algoritmuselmélet technikáit ötvözve optimalizálási feladatokat old meg struktúrák egy diszkrét halmazán. Egy kombinatorikus optimalizálási feladat megoldásakor tehát egy diszkrét halmazon értelmezett függvény optimumát keressük.

A struktúra halmaz elemei $0-1$ vektorok, és konkrét kombinatorikus optimalizációs feladatokban kombinatorikai struktúrákat reprezentálhatnak, mint például egy gráf köreit, vagy feszítőfáit. Ez elképzelhető például úgy, hogy a gráf minden lehetséges élének megfeleltetünk egy változót, ami 0-át vagy 1-et vehet fel, attól függően, hogy az adott él be van-e húzva a reprezentálandó gráfban, azaz minden n pontú gráf egyértelműen megfeleltethető egy $p = n(n-1)/2$ hosszú $0-1$ vektornak.

Mivel a halmaz nagysága exponenciálisan nőhet a reprezentált gráf, vagy egyéb objektum méretének függvényében, így az a lehetséges eljárás, ami egyenként megvizsgálja a feladat lehetséges megoldásait, általában gyakorlatilag nem működik. A lehetséges megoldások nagy mérete miatt egy ilyen eljárás még a leggyorsabb számítógépeken is évmilliókig tartana.

A cél tehát a gyakorlatban is jól használható, polinomiális, vagy erősen polinomiális algoritmusok megkonstruálása.

A kombinatorikus optimalizálás témakörébe tartozó kérdések már régóta igen érdekesek gyakorlati szempontból, és emellett egyre nagyobb igény van az olyan nagyméretű problémák megoldására, mint például távközlési hálózatok tervezése, tömegközlekedési eszközök menetrendjének optimalizálása, adótornyok, átjátszótornyok elhelyezésének optimalizálása, és egyéb ütemezési feladatok.

Emiatt létét tulajdonképpen a modern számítógépek megjelenésének köszönheti, hiszen a kombinatorikus optimalizálási feladatok megoldására kidolgozott algoritmusok igen sok számítást igényelnek, így számítógépek nélkül a gyakorlatban is előforduló problémák megoldására nem igazán alkalmazhatóak.

A szakdolgozat a kombinatorikus optimalizálás egy speciális témakörével, a lineáris kombinatorikus törtfüggvény optimalizációval, annak megoldási módszereivel, és alkalmazásaival foglalkozik. Mint az elnevezésből is látszik, ez a problémakör a kombinatorikus optimalizálás, a lineáris kombinatorikus optimalizálás, és a törtfüggvény optimalizálás témaköreibe is tartozik, pontosabban ezek speciális eseteit ötvözi, ezért mindegyik problémáról szó lesz röviden.

Ahol csak tehetjük az általánosabb, nem lineáris esetre mondjuk ki az állításokat, amelyek természetesen működnek a lineáris esetben is. Ha ki tudjuk használni a linearitást, hogy jobb becsléseket, vagy hatékonyabb algoritmusokat kapjunk, azt külön kiemeljük.

Egy lineáris kombinatorikus optimalizációs probléma általában a következőképpen írható le: az a cél, hogy találjunk egy minimális költségű kombinatorikus struktúrát, ahol a struktúra költsége az elemeinek összköltsége. Pontosabban leírva egy lineáris költségfüggvényt akarunk optimalizálni egy diszkrét halmaz felett.

Ilyen például a maximális költségű feszítőfa probléma, a legkisebb költségű kör keresésének problémája, illetve a minimális vágás megtalálásának problémája is. A struktúrák itt feszítőfái, körei vagy vágásai a gráfnak, a struktúrák elemei pedig a gráf éleinek felelnek meg.

Gyakran a költségek mellett súlyokat is kapnak az egyes elemek, és a feladat a maximális vagy minimális súlyozott átlagköltségű struktúra megtalálása. Egy struktúra súlyozott átlagköltsége: a költsége osztva a súlyával, ahol a súlya az elemeinek összsúlyával egyezik meg. Ezeket a feladatokat nevezzük lineáris kombinatorikus törtfüggvény optimalizálási problémának, esetleg 0-1 tört programozási problémának, vagy minimum-arány problémának.

A maximális súlyozott átlagköltségű feszítőfa probléma, a legkisebb súlyozott átlagköltségű kör keresésének problémája, illetve a minimális súlyozott átlagköltségű vágás megtalálásának problémája is mind példa a lineáris kombinatorikus törtfüggvény optimalizálási feladatra.

1.0.1. Példa. (Maximális súlyozott átlagköltségű feszítőfa keresése) Ebben a feladatban a struktúrák halmaza a gráf összes feszítőfájának halmaza. A $G = (V, E)$ gráf leírása egyben az $\mathcal{X} \subseteq \{0,1\}^m$ halmaz definíciója is, amely a G gráf összes feszítőfájának karakterisztikus vektorát tartalmazza ($m = |E|$).

Így tehát egy $\mathbf{x} = (x_1, ..., x_m) \in \mathcal{X}$ vektor egy G-beli feszítőfát reprezentál, amelyben az i. él pontosan akkor szerepel, ha $x_i = 1$.

Megadjuk még az $\mathbf{a} = (a_1, ..., a_m)$ és $\mathbf{b} = (b_1, ..., b_m)$ értékeket, ahol a_i az i. él költsége, és b_i az i. él súlya.

A feladat, hogy megtaláljuk a maximális átlagsúlyú költséget, és azt a feszítőfát (illetve az azt reprezentáló vektort), ahol ezt a maximális értéket felveszi.

2. fejezet

Kombinatorikus törtoptimalizálás

Ebben a fejezetben a legáltalánosabb esettől indulva ismertetjük a különböző kombinatorikus optimalizálási feladatokat. Elsőként az alapprobléma definíciója következik, majd ennek gyengébb változata, és lineáris változata. Ezek megoldása gyakran szükséges majd a későbbi feladatokban részfeladatként.

2.0.2. Definíció. *Az alapprobléma egy esete tartalmazza struktúrák egy halmazát:* $\mathcal{X} \subseteq \{0,1\}^p$*, és egy* $f : \mathcal{X} \to \mathbb{R}$ *függvényt. Az* $\mathcal{A}_{opt}$ *feladat:*

$$\max\{f(\boldsymbol{x}),\ ahol\ \boldsymbol{x} \in \mathcal{X}\}.$$

Sok esetben elég az alapprobléma gyengébb verzióját megoldani.

2.0.3. Definíció. *Az alapprobléma gyenge változata tartalmazza struktúrák egy halmazát:* $\mathcal{X} \subseteq \{0,1\}^p$*, és egy* $f : \mathcal{X} \to \mathbb{R}$ *függvényt. Az* $\mathcal{A}_{dec}$ *feladat:*

$$keressünk\ olyan\ \boldsymbol{y} \in \mathcal{X}\text{-}t,\ amelyre\ sign(f(\boldsymbol{y})) = sign\left(\max\{f(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}\}\right).$$

Mindkét esetben egy megfelelő $\mathbf{x}$-et akarunk találni, de amíg $\mathcal{A}_{opt}$ az optimális megoldást keresi, addig $\mathcal{A}_{dec}$ csak az optimális megoldás előjelét. A következő feladat az alapprobléma egy speciális esete, de mivel a lineáris feladatokban elég, ha csak ezt tudjuk megoldani, ezért külön is definiáljuk.

2.0.4. Definíció. *A lineáris alapprobléma tartalmaz egy* $\mathcal{X} \subseteq \{0,1\}^p$ *halmazt, és egy* $\boldsymbol{a} \in \mathbb{R}^p$ *vektort. Az* $\mathcal{A}_{max}$ *feladat:*

$$\max\{\boldsymbol{ax},\ ahol\ \boldsymbol{x} \in \mathcal{X}\}.$$

Az alapprobléma megoldásakor tehát struktúrák véges halmazán keressük az optimális struktúrá(ka)t, azaz egy költségfüggvény szerint optimalizálunk. Azonban gyakran nem csak egy költ-

ségfüggvény, hanem egy súlyfüggvény is adott, és a feladat a súlyozott átlagköltség maximalizálása. Egy tört optimalizálási feladatban tehát két adott függvény hányadosának optimumát akarjuk megtalálni.

2.0.5. Definíció. *Egy kombinatorikus törtfüggvény optimalizációs feladat tartalmazza struktúrák egy halmazát:* $\mathcal{X} \subseteq \{0,1\}^p$*, és két függvényt:* $g : \mathcal{X} \to \mathbb{R}$*,* $f : \mathcal{X} \to \mathbb{R}$*. Az* $\mathcal{F}$ *feladat:*

$$\max \quad \frac{f(x)}{g(x)}, \quad ahol \quad \boldsymbol{x} = (x_1, ...x_p) \in \mathcal{X}.$$

Feltesszük, hogy $f(\mathbf{x}) > 0$ valamely $\mathbf{x} \in \mathcal{X}$-re, és $g(\mathbf{x}) > 0$ minden $\mathbf{x} \in \mathcal{X}$-re. Az $f(\mathbf{x})$-et az $\mathbf{x}$ vektor által reprezentált struktúra költségének, $g(\mathbf{x})$-et a súlyának, $\frac{f(\mathbf{x})}{g(\mathbf{x})}$-et pedig a súlyozott költségének nevezzük.

A két legfontosabb és leghatékonyabb metódus a probléma megoldására Newton metódusa illetve Megiddo paraméteres kereső-metódusa. Mindkettő a törtfüggvény optimalizálásról a nem tört alakú, paraméteres optimalizálásra való visszavezetésen alapul.

2.0.6. Megjegyzés. A kombinatorikus törtfüggvény optimalizációs feladat speciális esete az általános törtfüggvény optimalizációs feladatoknak, amelyek $\frac{f(x)}{g(x)}$-et maximalizálják egy adott $D \subset \mathbb{R}^p$ részhalmaz felett.

Sokan tanulmányozták a problémát, és számos algoritmust fejlesztettek ki a feladat megoldására ([5], [6]). Ezek közül a numerikusak nem csak az általános, hanem a kombinatorikus esetben is használhatóak, de a két különböző esetben a metódusok vizsgálata és elemzése jelentősen különbözhet.

Például, amikor az általános törtfüggvény optimalizálási feladatot tekintjük, sokszor feltesszük, hogy a D tartomány konvex részhalmaza $\mathbb{R}^p$-nek, valamint az f és g függvények folytonosak. Világos, hogy ezeknek a feltételeknek nincs megfelelőjük a kombinatorikus esetben.

Továbbá, egy numerikus metódus az általános esetben konvergál ugyan az optimum értékhez, de nem tudjuk garantálni, hogy el is éri azt, míg a kombinatorikus esetben a megfelelő metódus általában kiadja az optimális megoldást. Vegyük észre, hogy a $\mathcal{X}$ halmaz véges, ezért az optimális megoldás megtalálható véges időben, egyszerűen végignézve minden $x \in \mathcal{X}$-et.

A Newton metódust az általános esetre Dinkelbach vezette be ([7]), és számos eredmény vonatkozik a metódus gyors konvergenciájára, és variációira is ([8], [9]).

Megiddo paraméteres kereső-metódusa olyan kombinatorikus törtfüggvény optimalizációs esetre készült, amikor a két függvény, f és g lineárisak. Ez a metódus volt az első például, amelyik erősen polinomiális megoldást adott a maximum profit/idő problémára, amikor a futási idő polinomiálisan függ p-től, de nem függ a célfüggvények értékeinek nagyságától.

2.0.7. Definíció. (Lineáris kombinatorikus törtfüggvény optimalizációs probléma) *Egy adott $\mathcal{F}_L$ lineáris kombinatorikus törtfüggvény optimalizációs feladat tartalmazza struktúrák egy halmazát: $\mathcal{X} \subseteq \{0,1\}^p$, és két valós, vagy egész vektort, az $\boldsymbol{a}$ költségfüggvényt, és a $\boldsymbol{b}$ súlyfüggvényt, ahol $\boldsymbol{a} = (a_1, ..., a_p)$ és $\boldsymbol{b} = (b_1, ..., b_p)$. Az $\mathcal{F}_L$ feladat :*

$$\max \quad \frac{a_1x_1 + ... + a_px_p}{b_1x_1 + ... + b_px_p}, \quad ahol \quad \boldsymbol{x} = (x_1, ...x_p) \in \mathcal{X}.$$

Az a_i és b_i számok az i. elem költsége illetve súlya, míg az $a_1x_1 + ... + a_px_p$, $b_1x_1 + ... + b_px_p$ és $\frac{a_1x_1+...+a_px_p}{b_1x_1+...+b_px_p}$ számok a költsége, súlya illetve súlyozott átlagköltsége a struktúrának, amit az $\mathbf{x}$ vektor reprezentál.

2.0.8. Jelölés. Egy $\mathbf{x}$ vektort a következőképpen jelölünk koordinátái segítségével: $\mathbf{x} = (x_1, x_2, ..., x_p)$.
Két egyenlő hosszú vektor ($\mathbf{a}$ és $\mathbf{x}$) skaláris szorzatát így jelöljük: $\mathbf{ax}$=$a_1x_1 + a_2x_2 + ... + a_px_p$.

Ezzel a jelöléssel a lineáris kombinatorikus törtfüggvény optimalizálási problémát így is felírhatjuk:

$$\mathcal{F}_L : \quad max \quad \frac{\mathbf{ax}}{\mathbf{bx}}, \quad ahol \quad \mathbf{x} \in \mathcal{X}$$

Feltesszük, hogy $\mathbf{bx} > 0$ minden $\mathbf{x} \in \mathcal{X}$-re, és van olyan $\mathbf{x} \in \mathcal{X}$, amelyre $\mathbf{ax} > 0$.

A következő tétel segítségével könnyen visszavezethetjük a kombinatorikus törtfüggvény optimalizálást nem tört alakú, paraméteres optimalizálásra.

2.0.9. Tétel. *Az $\mathcal{F}$ feladat megfogalmazható a következő ekvivalens módon:*

$$\mathcal{P} : \quad \min \{\delta : \ \delta \in \mathbb{R}, \ f(\boldsymbol{x}) - \delta g(\boldsymbol{x}) \leq 0, \quad minden \ \boldsymbol{x} \in \mathcal{X}\text{-}re\}.$$

Bizonyítás. Jelölje δ^* a minimumot, amit keresünk.
A $\mathcal{P}$ feladatban olyan δ-kat tekintünk, amelyekre $f(\mathbf{x}) - \delta g(\mathbf{x}) \leq 0$, azaz $\frac{f(\mathbf{x})}{g(\mathbf{x})} \leq \delta$ minden $\mathbf{x} \in \mathcal{X}$ esetén. Ha ezek közül a δ-k közül kiválasztjuk a legkisebbet, akkor (legalább) egy $\mathbf{x}^* \in \mathcal{X}$-re $\frac{f(\mathbf{x})}{g(\mathbf{x})} = \delta^*$, és minden $\mathbf{x} \in \mathcal{X} - \{\mathbf{x}^*\}$ esetén $\frac{f(\mathbf{x})}{g(\mathbf{x})} < \delta^*$, tehát megtaláltuk a $max \frac{f(\mathbf{x})}{g(\mathbf{x})}$ optimalizálási feladat megoldását. □

A $\mathcal{P}$ feladatot az $\mathcal{F}$ feladat paraméteres verziójának, vagy az $\mathcal{A}_{opt}$ probléma paraméteres kiterjesztésének is szokták nevezni.

A legtöbb iteratív megoldási módszer az $\mathcal{F}$ feladat megoldására a következő probléma különböző eseteinek sorozatát oldja meg, ahol a δ egy további input paraméter:

$$\mathcal{P}(\delta) : \quad \max_{x \in \mathcal{X}} \{f(x) - \delta g(x)\}.$$

A $\mathcal{P}(\delta)$ problémát az $\mathcal{F}$ feladathoz tartozó paraméteres problémának is szokták nevezni. Néhány megoldási módszerhez azonban elég a $\mathcal{P}(\delta)$ feladat gyengébb verzióját megoldani:

$\mathcal{P}_0(\delta)$: keressünk olyan $\mathbf{y} \in \mathcal{X} - et$, amelyre $sign(f(\mathbf{y}) - \delta g(\mathbf{y})) = sign(\max_{x \in \mathcal{X}} \{f(\mathbf{x}) - \delta g(\mathbf{x})\})$.

Tekintsük a következő segédfüggvényt tetszőleges $\delta \in \mathbb{R}$ esetén:

$$h(\delta) = \max_{x \in \mathcal{X}} \{f(\mathbf{x}) - \delta g(\mathbf{x})\}.$$

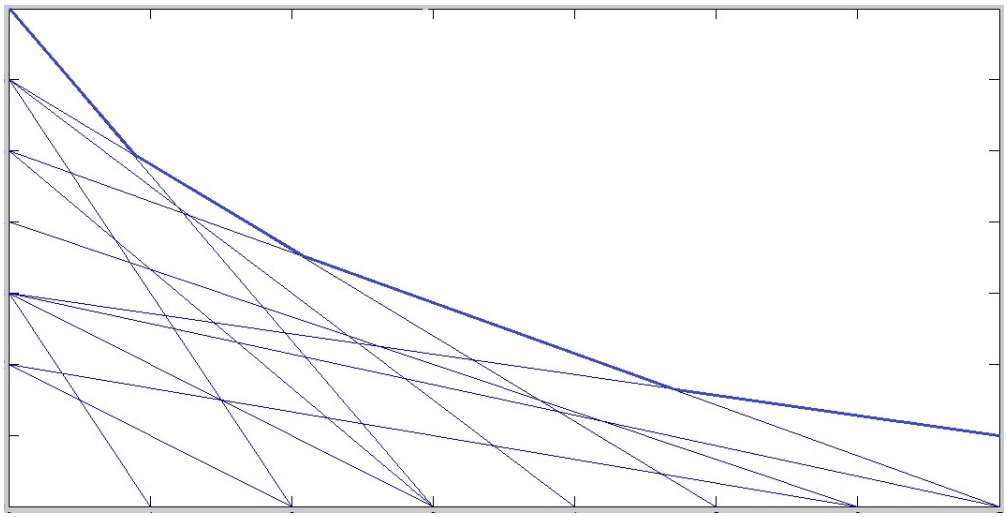
2.0.10. Megjegyzés. Ha a lineáris kombinatorikus törtfüggvény optimalizálási feladathoz tartozó $h(\delta)$ segédfüggvényt tekintjük, akkor az a következőképpen néz ki:

$$h(\delta) = \max_{x \in \mathcal{X}} \{(\mathbf{ax}) - \delta(\mathbf{bx})\} = \max_{x \in \mathcal{X}} \{(\mathbf{a} - \delta\mathbf{b})\mathbf{x})\},$$

Tetszőleges adott δ-ra az $\mathbf{a} - \delta\mathbf{b}$ vektort redukált költségnek nevezzük, $h(\delta)$-t pedig a struktúrák maximális redukált költségének.

2.0.11. Állítás. *A $h(\delta)$ függvény folytonos és szigorúan monoton csökken, konvex, szakaszonként lineáris, és egyetlen gyöke a δ^*.*

Bizonyítás. Minden rögzített x-re $f(x) - \delta g(x)$ lineáris, csökkenő függvénye δ-nak, mert $f(x)$ és $g(x)$ adott számok, és $g(x) > 0$ a feltevésünk miatt. Tehát a h függvény véges sok lineáris függvény maximuma, ezért szakaszosan lineáris. Mivel ezek a lineáris függvények mind szigorúan monoton csökkenek, ezért a h függvény is szigorúan monoton csökken. Mivel azt is tudjuk, hogy nincs olyan x, amelyre $g(x) < 0$, ezért az is nyilvánvaló, hogy a h konvex, és pontosan egy gyöke van. □



2.1. ábra. Példa a h függvényre

Ezzel a kombinatorikus törtfüggvény optimalizációs probléma egy újabb ekvivalens megfogalmazását kapjuk:

$$\mathcal{R} : \text{oldjuk meg a } h(\delta) = 0 \text{ feladatot.}$$

Ugyanis, ha

$$h(\delta^*) = \max_{x \in \mathcal{X}} \{f(\mathbf{x}) - \delta^* g(\mathbf{x})\} = 0,$$

akkor az pont azt jelenti, hogy $f(\mathbf{x}) - \delta^* g(\mathbf{x}) \leq 0$ minden $\mathbf{x} \in \mathcal{X}$-re, de létezik $\mathbf{x}^* \in \mathcal{X}$, amire $f(\mathbf{x}^*) - \delta^* g(\mathbf{x}^*) = 0$, azaz $\mathbf{x}^*$-nál veszi fel a maximumát az $\frac{f(\mathbf{x})}{g(\mathbf{x})}$.

Ez a megfogalmazás azt sugallja, hogy az $\mathcal{F}$ probléma megoldására egyszerűen olyan algoritmusokat készíthetünk, amelyek egy függvény gyökét keresik.

2.0.12. Definíció. *Az uniform lineáris kombinatorikus törtfüggvény optimalizálási probléma:*

$$\mathcal{U} : \quad \max \quad \frac{\boldsymbol{ax}}{\boldsymbol{ex}}, \quad ahol \quad \boldsymbol{x} \in \mathcal{X}.$$

Az $\boldsymbol{e}$ vektor a következő: $\boldsymbol{e} = (1, 1, ..., 1)$. A feladat itt a maximális átlagköltségű struktúra megtalálása. Egy struktúra átlagköltsége a költsége osztva az elemszámával.

Nevezik még homogén vagy súlyozatlan esetnek.

3. fejezet

Megoldási módszerek

3.1. Bináris keresés a lineáris kombinatorikus törtfüggvény optimalizációra

A bináris keresés a lineáris kombinatorikus törtfüggvény optimalizációs feladatok megoldására egyszerűen a bináris keresés alkalmazása $h(\delta)$ gyökének, azaz δ^* megtalálására. Minden iterációban a lineáris alap optimalizáció egy esetét oldjuk meg, hogy kiszámítsuk az aktuális δ közelítésre $h(\delta)$ előjelét.

Legyen a kezdő közelítő intervallum $[\xi_1, \lambda_1]$, amelyről tudjuk, hogy $h(\xi_1) > 0$, és $h(\lambda_1) < 0$. Ez biztosítja, hogy δ^* az intervallumon belül van. Minden iterációban felezzük az intervallumot, $\psi_i = \frac{\lambda_i - \xi_i}{2} + \xi_i = \frac{\xi_i + \lambda_i}{2}$, és kiszámítjuk $h(\psi_i)$ előjelét. Ha pozitív, akkor $\psi_i < \delta^*$, ekkor $\xi_{i+1} = \psi_i$ és $\lambda_{i+1} = \lambda_i$, ha negatív, akkor $\psi_i > \delta^*$, és ekkor $\lambda_{i+1} = \psi_i$ és $\xi_{i+1} = \xi_i$ lesz az új közelítés, az új intervallum, amelyik fele akkora, és tartalmazza δ^*-t.

3.1.1. Megjegyzés. Fontos kiemelnünk, hogy a bináris keresés alkalmazásakor a lineáris alap optimalizációs feladat "gyenge" verzióját, $\mathcal{A}_{dec}$-t kell megoldanunk részfeladatként, míg a Newton-metódushoz az "erősebb" változat, $\mathcal{A}_{opt}$ kell, hogy találjunk egy struktúrát a maximális redukált költséggel, és ez a különbség néha drasztikus lehet.
Tekintsük például a maximum átlagsúlyú kör keresésének problémáját. A bináris kereséshez egy olyan eljárás szükséges, amelyik megmondja van-e pozitív redukált költségű kör a gráfban, ami $O(mn)$ időben kiszámolható. Ezzel szemben a Newton-metódusnak egy olyan eljárásra van szüksége, amelyik meghatározza a maximális költségű kört, ami NP-nehéz feladat.

Ha a lineáris esetben a költség-, és súlyfüggvények egészértékűek, és $a_i \in [-A, A]$, $b_i \in [1, B]$, akkor könnyű becslést adni az iterációk számára. Vizsgáljuk meg, hogy legalább mekkora kell

legyen két lehetséges tört-optimum között a különbség:

$$\left|\frac{\mathbf{ax}}{\mathbf{bx}} - \frac{\mathbf{ay}}{\mathbf{by}}\right| = \left|\frac{\mathbf{ax}\cdot\mathbf{by} - \mathbf{ay}\cdot\mathbf{bx}}{\mathbf{bx}\cdot\mathbf{by}}\right| \geq \left|\frac{1}{(pB)^2}\right|.$$

A kezdő közelítő intervallum legyen $[-pA, pA]$, hiszen $|\mathbf{ax}| \leq pA$, és $|\mathbf{bx}| \geq 1$, ezért $\left|\frac{\mathbf{ax}}{\mathbf{bx}}\right| \leq pA$ tetszőleges $\mathbf{x}$ esetén. Ha k iterációt végzünk el, akkor az $[\xi_k, \lambda_k]$ intervallum hossza $\frac{2\cdot pA}{2^k}$.

Ezért, ha azt akarjuk, hogy a k. iteráció után olyan intervallumot kapjunk, amiben már csak a δ^* optimum szerepel, a következőnek kell teljesülnie:

$$\frac{2\cdot pA}{2^k} < \frac{1}{(pB)^2}.$$

$$2^k > \frac{2\cdot pA}{\frac{1}{(pB)^2}} = 2p^3AB^2.$$

$$k > \log(2p^3AB^2)$$

Jelöljük U-val az A és B számok maximumát.

$$k > \log(2p^3U^3) = \log 1 + 3\log(pU) = O(\log(pU)).$$

Tehát $O(\log(pU))$ lépésben olyan intervallumot kapunk, amelyben már csak az optimális megoldás szerepel, és semmilyen másik $\frac{ay}{by}$ érték nincs benne. Ez azt jelenti, hogy ha ekkor kiszámítjuk az utolsó iterációval kapott intervallum baloldali határának, ξ_k-nak a maximális redukált költségét, azaz $h(\xi_k)$-t, és a hozzá tartozó x_k vektort, ahol ez a maximális költség felvétetik, akkor megkaptuk azt az "utolsó" $(\mathbf{ax}_k) - \delta(\mathbf{bx}_k)$ egyenest, amelynek az x-tengellyel vett metszési pontja már meghatározza az optimumot.

Összesen tehát $O(\log(pU))$ alkalommal kell megoldanunk az $\mathcal{A}_{dec}$ feladat egy esetét, és ezen kívül egyszer az $\mathcal{A}_{opt}$ feladatot.

3.2. Newton metódusa a lineáris kombinatorikus törtfüggvény optimalizálásra

Newton metódusa a lineáris kombinatorikus törtfüggvény optimalizálásra nem más, mint Newton metódusa $h(\delta)$ zérushelyének megtalálására. Mivel Dinkelbach használta először tört optimalizálásra, ezért Dinkelbach metódusnak is nevezik. Ebben a részben bemutatjuk a Newton metódust, mint megoldási módszert az általánosabb kombinatorikus törtfüggvény optimalizálásra. Ezen kívül a lineáris esetben mutatunk erősen polinomiális becslést is az iterációk számára, és még jobb becslést egész célfüggvények esetében.

A Newton metódus olyan iteratív eljárás az $\mathcal{F}$ feladatra, amely minden iterációs lépésben egyre jobb tört megoldást, és egyre jobb becslést ad a δ^* optimumértékre is.

Hogy használni tudjuk Newton metódusát, fel kell tennünk, hogy van egy eljárás, ami megoldja $\mathcal{A}_{opt}$-t. Ezt fogjuk használni, hogy bármely adott δ-ra kiszámíthassuk $h(\delta)$-t, a struktúrák maximális redukált költségét, és magát az $\mathbf{x} \in \mathcal{X}$ struktúrát, ahol ez a költség felvétetik.

Legyen $\bar{\delta}$ az aktuális közelítése δ^*-nak. Kezdetben $\bar{\delta} = 0$. Feltevéseink biztosítják, hogy $\delta^* > 0$, ugyanis $g(\mathbf{x}) > 0$ minden $\mathbf{x} \in \mathcal{X}$-re, és van olyan $\mathbf{x} \in \mathcal{X}$, amelyre $f(\mathbf{a}) > 0$. Egy iteráció részeként kiszámítjuk $h(\bar{\delta})$-t, és azt az $\bar{\mathbf{x}} \in \mathcal{X}$ vektort, amelyen a maximum felvétetik, azaz $h(\bar{\delta}) = f(\bar{\mathbf{x}}) - \bar{\delta} g(\bar{\mathbf{x}})$. Ehhez szükséges az $\mathcal{A}_{opt}$ probléma egy esetének megoldása. Ha $h(\bar{\delta}) = 0$, akkor $\delta^* = \bar{\delta}$, és az algoritmus véget ér. Máskülönben kiszámítjuk a következő közelítést: $\bar{\delta}' = f(\bar{\mathbf{x}})/g(\bar{\mathbf{x}})$, ami az $\bar{\mathbf{x}}$ vektor súlyozott átlagköltsége, és az algoritmus folytatódik az új közelítéssel számolva az előbb leírt módon.
Azért így számoljuk a következő közelítés értéket, mert azt a helyet keressük, ahol $h(\bar{\delta}') = 0$ (ugyanis a Newton-módszer szerint az új közelítés ott lesz, ahol az előző közelítés függvényértékéből induló érintő metszi az x tengelyt), tehát ahol $f(\bar{\mathbf{x}}) - \bar{\delta}' g(\bar{\mathbf{x}}) = 0$, azaz $\frac{f(\bar{\mathbf{x}})}{g(\bar{\mathbf{x}})} = \bar{\delta}'$.

3.2.1. Jelölés. Legyen δ_i $\bar{\delta}$ értéke az i. iteráció elején, és x_i, h_i, g_i legyenek $\bar{\mathbf{x}}$, $f(\bar{\mathbf{x}}) - \delta_i g(\bar{\mathbf{x}})$ illetve $g(\bar{\mathbf{x}})$ szintén az i. iteráció kezdetekor. Így

$$h_i = h(\delta_i) = f(\mathbf{x}_i) - \delta_i g(\mathbf{x}_i) = max\left\{f(\mathbf{x}) - \delta_i g(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\right\}$$

$$g_i = g(\mathbf{x}_i)$$
$$\delta_{i+1} = \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)}$$

3.2.2. Megjegyzés. A lineáris esetben legyenek a következők a jelölések: Legyen δ_i és x_i $\bar{\delta}$, illetve $\bar{\mathbf{x}}$ értéke az i. iteráció elején, és

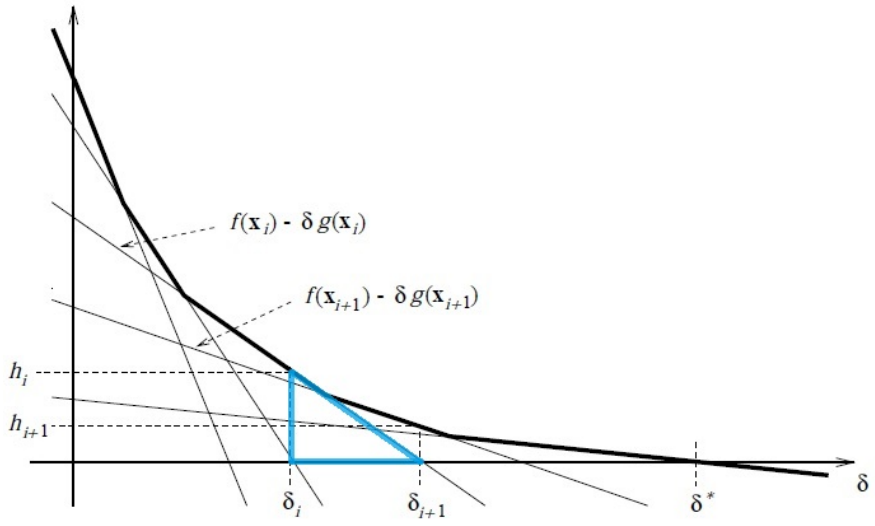
$$H_i = h(\delta_i) = (\mathbf{a} - \delta_i \mathbf{b})\mathbf{x}_i = max\left\{(\mathbf{a} - \delta_i \mathbf{b})\mathbf{x} : \mathbf{x} \in \mathcal{X}\right\}$$

$$B_i = \mathbf{b}\mathbf{x}_i$$
$$\delta_{i+1} = \frac{\mathbf{a}\mathbf{x}_i}{\mathbf{b}\mathbf{x}_i}$$

Könnyen igazolható a következő állítás:

$$\delta_{i+1} - \delta_i = \frac{h_i}{g_i}. \tag{3.1}$$

A $\delta_{i+1} = \frac{f(x_i)}{g(x_i)}$ összefüggésből és a h_i definíciójából valóban egyszerűen következik.
Kicsit szemléletesebben is beláthatjuk: ha a (3.1) ábrán megnézzük a kék színnel kiemelt háromszöget, észrevehető, hogy a háromszög egyik befogója éppen $\delta_{i+1} - \delta_i$, másik befogója h_i, és a háromszög alapon fekvő szögének tangense $\frac{h_i}{\delta_{i+1} - \delta_i}$, ami az $f(x_i) - \delta g(x_i)$ egyenes iránytangensének a mínusz egyszerese, ami pedig az egyenes meredeksége, azaz $-g_i$, tehát $-g_i = -\frac{h_i}{\delta_{i+1} - \delta_i}$.



3.1. ábra. Az algoritmus egy lépése
[3]

A célunk becslést adni az iterációk számára. Legyen t az utolsó iteráció indexe (vagy ∞, ha az eljárás nem áll le), és jelöljük f_i-vel $f(\mathbf{x}_i)$-t, ha $i = 1, ..., t$. A korábbiak szerint $h_i = f_i - \delta_i g_i$, és azt is tudjuk, hogy $\delta_{i+1} = \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)}$.

3.2.3. Lemma. *A Newton metódus véges sok lépés után véget ér, és*

1. $h_1 > h_2 > ... > h_{t-1} > h_t = 0$,
2. $0 = \delta_1 < \delta_2 < ... < \delta_{t-1} < \delta_t = \delta^*$,
3. $g_1 > g_2 > ... > g_{t-1} > g_t$.

Bizonyítás. A lemma következik az alábbi állításokból: minden $1 \leq i < t + 1$-re

(a) $h_1 > h_2 > ... > h_{i-1} > h_i \geq 0$,
(b) $0 = \delta_1 < \delta_2 < ... < \delta_{i-1} < \delta_i \leq \delta^*$,
(c) $g_1 > g_2 > ... > g_{i-1} \geq g_i$.

Az egyenlőtlenség láncok végén csak akkor fordulhat elő egyenlőség, ha az i. iteráció a metódus utolsó iterációja, tehát ha $i = t$.
Az előbbi állításokat i szerinti indukcióval bizonyítjuk. Mivel a h függvény tulajdonságaiból következik, hogy $h(0) > 0$, és az egyetlen gyöke pozitív, ezért $i = 1$-re igazak az állítások. Tegyük

tehát fel, hogy $i = j$-ig igazak az állítások, valamely $1 \leq j < t$-re. Tudjuk a következőket:

$$\delta^* \geq \frac{f_j}{g_j} = \delta_{j+1}, \quad \delta_{j+1} = \frac{f_j}{g_j} = \frac{h_j}{g_j} + \delta_j > \delta_j,$$

ezért a (b) állítás igaz $i = j + 1$-re. Mivel $\delta_j < \delta_{j+1} \leq \delta^*$, és tudjuk, hogy a h függvény monoton csökken, ezért $h_{j+1} = h(\delta_{j+1}) < h(\delta_j) = h_j$, és $h_{j+1} = h(\delta_{j+1}) \geq h(\delta^*) = 0$, ezért az (a) rész is igaz $i = j + 1$-re.
A (c) állítás igazolásához a következő két egyenlőtlenséget használjuk:

$$f_{j+1} - \delta_j g_{j+1} \leq max\,\{f_k - \delta_j g_k\} = h_j = f_j - \delta_j g_j,$$

$$f_{j+1} - \delta_{j+1} g_{j+1} = h_{j+1} = max\,\{f_k - \delta_{j+1} g_k\} \geq f_j - \delta_{j+1} g_j.$$

A két egyenlőtlenséget egymásból kivonva kapjuk, hogy $g_{j+1} \leq g_j$, és egyenlőség csak akkor állhat fenn, ha $h_{j+1} = f_j - \delta_{j+1} g_j$. Mivel $f_j - \delta_{j+1} g_j = 0$ a δ_{j+1} definíciója miatt, ezért $g_{j+1} = g_j$ csak akkor lehetséges, ha $h_{j+1} = 0$, ami viszont azt jelenti, hogy az $j + 1$. iteráció az utolsó. Ezzel igazoltuk a (c) állítást is.
Mindezek miatt a g_i sorozat elemei mind különbözőek, ezért a Newton iterációnak véges sok iteráció után véget kell érnie, mert a g_i számok a struktúrák súlyai, tehát $t - 1 \leq |\mathcal{X}| < \infty$. □

Az algoritmus gyors konvergenciájának bizonyításához szükséges tételt általánosabban mondjuk ki és bizonyítjuk, mégpedig tetszőleges konvex függvényre, és nem csak szakaszosan lineárisra, mint a h függvény. Ehhez azonban szükség lesz a differenciálhatóság fogalmának kiterjesztésére, mert nem tehetjük fel, hogy a függvényünk mindenhol differenciálható.

3.2.4. Definíció. *Legyen $S \subseteq \mathbb{R}$ nemüres halmaz, és legyen $f : S \to \mathbb{R}$ konvex függvény, $x_0 \in S$. A $g(x_0) \in \mathbb{R}$ számot az f függvény x_0 pontbeli szubgradiensének nevezzük, ha*

$$f(x) \geq f(x_0) + g(x_0)(x - x_0)\, minden\, x \in S - re.$$

3.2.5. Definíció. *Legyen $S \subseteq \mathbb{R}$ nemüres halmaz, és legyen $f : S \to \mathbb{R}$ konvex függvény, $x_0 \in S$. Az f függvény szubdifferenciálható az x_0 pontban, ha x_0-nak van legalább egy szubgradiense. Az x_0 pontbeli szubgradiensek halmazát szubdifferenciálnak nevezzük, és $\partial f(x_0)$-al jelöljük.*

3.2.6. Megjegyzés. Ha az $f : S \to \mathbb{R}$ függvény differenciálható az $x_0 \in S$ pontban, akkor pontosan egy szubdifferenciálja van, ami pont $f'(x_0)$.

3.2.7. Állítás. *Legyen $S \subset \mathbb{R}$ nemüres halmaz, és legyen $f : S \to \mathbb{R}$ folytonos konvex függvény. Ekkor $\partial f(x)$ nemüres és korlátos.*

3.2.8. Jelölés. Legyen $S \subset \mathbb{R}$ nemüres halmaz, és $f : S \to \mathbb{R}$ folytonos konvex függvény. Legyen ekkor $f'(x)$ a egy tetszőleges szubgradiens $\partial f(x)$-ből minden $x \in S$-re.

3.2.9. Állítás. *Legyen $f : \mathbb{R} \to \mathbb{R}$ folytonos konvex függvény, amiről tudjuk, hogy $\exists\, x^* : f(x^*) = 0$. Legyen x_0 kezdő közelítés, amelyre $f(x_0) > 0$, és $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ a Newton-metódus szabálya szerint. Ekkor*

$$\frac{f(x_{i+1})}{f(x_i)} + \frac{f'(x_{i+1})}{f'(x_i)} \leq 1$$

.

Bizonyítás. Mivel f konvex, ezért felírhatjuk tetszőleges $x_i, x_{i+1} \neq x^*$, $x_i < x_{i+1}$-re a következőt:

$$f(x_i) \geq f'(x_{i+1})(x_i - x_{i+1}) + f(x_{i+1})$$

Ha x_{i+1} helyére behelyettesítjük a feltételben szereplő kifejezés jobb oldalát, a következőt kapjuk:

$$f(x_i) \geq f'(x_{i+1})\big(x_i - (x_i - \frac{f(x_i)}{f'(x_i)})\big) + f(x_{i+1})$$

Azaz:

$$f(x_i) \geq f'(x_{i+1})\frac{f(x_i)}{f'(x_i)} + f(x_{i+1})$$

Osszunk le $f(x_i)$-vel.
(Megtehetjük, mert tudjuk, hogy $f(x_0) > 0$, és x_i nem gyök, tehát $f(x_i) > 0$.) Így kapjuk:

$$1 \geq \frac{f'(x_{i+1})}{f'(x_i)} + \frac{f(x_{i+1})}{f(x_i)}.$$

□

Az előbbi tétel közvetlen következménye az alábbi egyenlőtlenség, amit az algoritmus gyors konvergenciájának bizonyításához fogunk felhasználni:

$$\frac{h_{i+1}}{h_i} + \frac{g_{i+1}}{g_i} \leq 1. \tag{3.2}$$

A későbbiekben hasznos lesz az (3.2)-as egyenlőtlenség alábbi egyszerű következménye, amelyet a számtani-mértani közepek közti egyenlőtlenséget használva kaphatunk meg:

$$\frac{h_{i+1}g_{i+1}}{h_i g_i} \leq \frac{1}{4}. \tag{3.3}$$

3.2.10. Megjegyzés. A lineáris esetben tehát (3.2), illetve (3.3) egyenlőtlenségek a következőképpen néznek ki:

$$\frac{H_{i+1}}{H_i} + \frac{B_{i+1}}{B_i} \leq 1. \tag{3.4}$$

$$\frac{H_{i+1}B_{i+1}}{H_i B_i} \leq \frac{1}{4}. \tag{3.5}$$

3.2.1. Polinomiális lépésszám egészértékű célfüggvények esetében

Ebben a részben csak a lineáris esettel foglalkozunk, és feltesszük, hogy mind a költségek, mind a súlyok egész számok. Legyen A és B a költségek illetve súlyok abszolútértékeinek maximuma, illetve jelölje U az A és B szám maximumát.

Az (3.5) egyenlőtlenség következményeképpen igazolható a következő polinomiális lépésszám a Newton metódus esetén.

3.2.11. Tétel. *A Newton metódus egy lineáris kombinatorikus törtfüggvény optimalizációs problémát $O(\log(pU))$ lépésben old meg.*

Bizonyítás. Ha $H_i > 0$ akkor minden i-re:

$$(pU)^2 \geq H_i B_i \geq \frac{1}{pU}. \tag{3.6}$$

Az első egyenlőtlenség abból következik, hogy

$$H_i = h(\delta_i) \leq h(0) = \max_{\mathbf{x}\in\mathcal{X}} \{(\mathbf{a} - 0\mathbf{b})\mathbf{x}\} = max_{\mathbf{x}\in\mathcal{X}} \{\mathbf{ax}\} \leq pU.$$

A második egyenlőtlenség pedig a következők miatt igaz:

$$H_i = (\delta_{i+1} - \delta_i)B_i = \frac{(\mathbf{ax}_i)(\mathbf{bx}_{i-1}) - (\mathbf{ax}_{i-1})(\mathbf{bx}_i)}{(\mathbf{bx}_i)(\mathbf{bx}_{i-1})} B_i \geq \frac{1}{B_i B_{i-1}} B_i \geq \frac{1}{pU}. \tag{3.7}$$

A (3.5) és (3.6) egyenlőtlenségekből már következik a tétel:

$$\frac{1}{pU} \leq H_k B_k \leq (\frac{1}{4})^{k-1}(pU)^2$$

$$4^{k-1} \leq (pU)^3$$

$$k - 1 \leq \frac{1}{\log 4} 3 \log pU$$

□

Most következik az előbbi tétel finomítása, ami jobb lépésszámot biztosít, feltéve, hogy a súlyfüggvény értékei lényegesen kisebbek, mint a költségfüggvényé. Ez tipikusan az uniform lineáris kombinatorikus törtfüggvény optimalizációs probléma esetében fordulhat elő, ahol a súlyfüggvény azonosan egy.

Szükségünk lesz a következő lemmára.

3.2.12. Lemma. *Legyenek μ, α és β pozitív számok, $\mu < 2$, $\alpha < 1$, $\beta < 1$. Legyenek (α_i) és (β_i) pozitív tagú, l hosszú sorozatok, úgy hogy*

$$\alpha_i + \beta_i \leq \mu \quad (i = 1, 2, ..., l),$$

$$\prod_{1\le i\le l}\alpha_i \ge \alpha, \quad \prod_{1\le i\le l}\beta_i \ge \beta.$$

Ekkor $l \le L$, ahol L a következő egyenlőtlenség megoldása:

$$\alpha^{\frac{1}{L}} + \beta^{\frac{1}{L}} = \mu.$$

Bizonyítás. A számtani-mértani közepek közti egyenlőtlenséget használva:

$$\alpha^{\frac{1}{l}} + \beta^{\frac{1}{l}} \le \Big(\prod_{1\le i\le l}\alpha_i\Big)^{\frac{1}{l}} + \Big(\prod_{1\le i\le l}\beta_i\Big)^{\frac{1}{l}} \le \frac{1}{l}\Big(\sum_{1\le i\le l}\alpha_i\Big) + \frac{1}{l}\Big(\sum_{1\le i\le l}\beta_i\Big) = \frac{1}{l}\Big(\sum_{1\le i\le l}(\alpha_i+\beta_i)\Big) \le \mu$$

Ebből már következik, hogy $l \le L$, mert az $\alpha^{\frac{1}{x}} + \beta^{\frac{1}{x}}$ monoton növekszik a $(0,\infty)$-en. □

3.2.13. Tétel. *A Newton metódus egy lineáris kombinatorikus törtfüggvény optimalizációs problémát $O(\frac{\log(pAB)}{1+\log\log(pAB)-loglog(pB)})$ lépésben old meg.*

Bizonyítás. Feltesszük, hogy $B \le A$, ugyanis, ha $A > B$, akkor ugyanazt a becslést kapjuk, mint az előző tételben.

Jelölje l az iterációs lépések számát. Legyen $\alpha_i = \frac{H_{i+1}}{H_i}$, $\beta_i = \frac{B_{i+1}}{B_i}$, ahol $i = 1, 2, ..., l-1$.

A (3.4) egyenlőtlenségből tudjuk, hogy $\alpha_i + \beta_i < 1$, és igazak a következő becslések is:

$$H_1 \le pA, \quad H_l \ge \frac{1}{pB}, \quad B_1 \le pB, \quad B_l \ge 1.$$

A második egyenlőtlenség a (3.4) egyenlőtlenségből következik, a többi triviális. Ezeket felhasználva kapjuk, hogy

$$\prod_{1\le i\le l-1}\alpha_i = \frac{H_l}{H_i} \ge \frac{1}{p^2AB}, \quad \textit{illetve} \prod_{1\le i\le l-1}\beta_i = \frac{H_l}{H_i} \ge \frac{1}{pB}.$$

A 3.2.12 lemma miatt $l-1 \le L$, ahol L a következő egyenlet megoldása:

$$\left(\frac{1}{p^2AB}\right)^{\frac{1}{L}} + \left(\frac{1}{pB}\right)^{\frac{1}{L}} = 1.$$

Legyen $h(x) = \left(\frac{1}{p^2AB}\right)^{\frac{1}{x}} + \left(\frac{1}{pB}\right)^{\frac{1}{x}}$, *minden* $x > 0$-ra.

Ekkor

$$h\left(\frac{2\log(p^2AB)}{\log\log(p^2AB)-\log\log(pB)}\right) = \left(\frac{\log(pB)}{\log(p^2AB}\right)^{\frac{1}{2}} + \left(\frac{\log(pB)}{\log(p^2AB)}\right)^{\frac{\log(pB)}{2\log(p^2AB)}} > 1.$$

Az utóbbi egyenlőtlenség azért igaz, mert $z^{\frac{1}{2}} + z^{\frac{z}{2}} > 1$, ha $z > 0$, ami az elemi analízis eszközeivel igazolható.

A $h(x)$ függvény monoton nő, ezért

$$L < \frac{2\log(p^2AB)}{\log\log(p^2AB)-\log\log(pB)} = O\left(\frac{\log(pAB)}{1+\log\log(pAB)-\log\log(pB)}\right).$$

Mivel feltettük, hogy $B \le A$, ezért a fenti becslés igaz. □

3.2.14. Megjegyzés. Könnyen látható, hogy ezek a becslések nem csak egészértékű súly-, és költségfüggvények esetén alkalmazhatóak. Ha **a** és **b** racionális értékűek, és megkeressük azt a legkisebb c számot, amivel felszorozva mindkét vektort (így kapjuk **a'**-t és **b'**-t), azok értékei egészek lesznek, akkor

$$max \quad \frac{\mathbf{ax}}{\mathbf{bx}} = \quad max \quad \frac{a_1x_1 + a_2x_2 + ... + a_px_p}{b_1x_1 + b_2x_2 + ... + b_px_p} = \quad max \quad \frac{c \cdot (a_1x_1 + a_2x_2 + ... + a_px_p)}{c \cdot (b_1x_1 + b_2x_2 + ... + b_px_p)}$$

$$= \quad max \quad \frac{c \cdot a_1x_1 + c \cdot a_2x_2 + ... + c \cdot a_px_p}{c \cdot b_1x_1 + c \cdot b_2x_2 + ... + c \cdot b_px_p} = \quad max \quad \frac{\mathbf{a'x}}{\mathbf{b'x}}.$$

Tehát tetszőleges racionális értékű súly-, és költségfüggvény esetén igazak az előbbi becslések, mindössze annyit kell tenni, hogy felszorozzuk mindkettőt egy alkalmas egész számmal. Persze ekkor U értéke is változik: $U' = c \cdot U$, ezért $O(\log(pU')) = O(\log(pcU))$ lesz az új becslés az iterációk számára.

3.2.2. Erősen polinomiális lépésszám az általános esetben

Az általános eset tanulmányozása előtt az uniform lineáris kombinatorikus törtfüggvény optimalizálási problémával foglalkozunk, amikor a költségek tetszőleges valós számok, a súlyfüggvény pedig azonosan egy. A következő tényt először Karzanov publikálta [10].

3.2.15. Tétel. *Newton metódusa legfeljebb $p + 1$ lépésben megoldja az uniform lineáris kombinatorikus törtfüggvény optimalizálási problémát.*

Bizonyítás. A (B_i) sorozat szigorúan csökken, kivéve az utolsó iterációt. Az egységes lineáris kombinatorikus törtfüggvény optimalizálási problémában B_i pozitív egész, és legfeljebb p, ugyanis pont az adott struktúra számosságát adja meg. □

A következőkben erősen polinomiális becslést adunk a liníaris kombinatorikus törtfüggvény optimalizáció általános esetére, amikor a költségek, és a súlyok is tetszőleges valós számok.

Először nézzük meg, mi lehet az elmélet mögött. A (3.4) egyenlőtlenség azt sugallja, hogy vannak olyan legalább mértani gyorsasággal csökkenő sorozatok (pl. H_iB_i), amelyeknek a metódus gyorsaságához van köze (lásd (3.5)).

Hogy még inkább kiterjeszthessük az intuíciót, tegyük fel, hogy létezik olyan konstans α szám, amire igaz, hogy $\frac{B_{i+1}}{B_i} \leq \alpha < 1$ minden i-re. Ez azt jelenti, hogy a (B_i) sorozat legalább mértani gyorsasággal tart nullához. Tegyük fel azt is, hogy $b_1 \geq b_2 \geq ... \geq b_p \geq 0$. A (B_i) sorozat minden eleme a $\{b_1, b_2, ..., b_p\}$ halmaz egy részhalmazának elemeinek összege.
Nyilván $B_1 \leq pb_1$. Mivel (B_i) legalább egy mértani sorozat sebességével csökken, ezért $B_l \leq b_1$, ahol $l = O(\log p)$. Ez azt jelenti, hogy b_1 már nem lehet benne B_l-ben, sem semelyik B_i-ben, ha $i \geq l$, ezért innentől eltekinthetünk b_1-től.

Tehát $B_l \leq (p-1)b_2$, és a következő $O(\log p)$ iteráció után már b_2 sem szerepelhet a sorozat további tagjaiban, és hasonlóan sorban kizárhatjuk b_3-at, b_4-et, stb.
Ebből következően a (B_i) sorozat legfeljebb $O(p \log p)$ hosszú lehet, tehát legfeljebb ennyi iterációs lépést tehetünk.

Az iterációk számára vonatkozó becslés valójában bonyolultabb, mint ahogy azt az előző bekezdés sugallja, mivel a $(H_i B_i)$ számoknak "bonyolultabb a szerkezetük" mint a (B_i) sorozat tagjainak. Két oka van, hogy az általános eset bonyolultabb. Először is, a pozitív számok mellett a negatívakkal is számolnunk kell, ugyanis még ha minden súly és költség pozitív is, megjelennek a negatív számok, mert a (H_i) sorozat elemeinek számolásakor kivonást is használunk. Másodszor, ha a (B_i) sorozat nem csökken elég gyorsan, akkor a (H_i) sorozatot is meg kell vizsgálnunk, aminek az elemei azonban nem csupán egy előre meghatározott halmaz részhalmazainak elemeinek összege.
A következő lemma segítségével már fogunk tudni a negatív számokkal is bánni, ugyanis azt mondja ki, hogy még ha az alaphalmaz tartalmaz is pozitív és negatív számokat is, akkor is ezeknek a számoknak a részletösszegeiből képzett mértani sorozat hossza legfeljebb $O(p \log p)$.
A lemmát Michel Goemans mondta ki és bizonyította. Az állításban szereplő $\mathbf{c}$ vektor koordinátái segítségével képezzük a részletösszegeket, és az $(\mathbf{y}_k \mathbf{c})$ sorozat pedig a belőlük képzett sorozat.

3.2.16. Lemma. (Michel Goemans) *Legyen $\boldsymbol{c} = (c_1, c_2, ..., c_p)$ pozitív valós koordinátákból álló vektor, illetve $\boldsymbol{y}_1, \boldsymbol{y}_2, ... \boldsymbol{y}_q \subseteq \{0, 1, -1\}^p$.*

Ha minden $i = 1, 2, ... q-1$-re

$$0 < \boldsymbol{y}_{i+1} \boldsymbol{c} \leq \frac{1}{2} \boldsymbol{y}_i \boldsymbol{c},$$

akkor $q = O(p \log p)$.

Bizonyítás. Tekintsük a következő poliédert:

$\mathbf{z} = (z_1, z_2, ..., z_p) \in \mathbb{R}^p$:

$$(\mathbf{y}_i - 2\mathbf{y}_{i+1})\mathbf{z} \geq 0, \quad i = 1, 2, ..., q-1-re \tag{3.8}$$

$$\mathbf{y}_q \mathbf{z} = 1, \tag{3.9}$$

$$\mathbf{z}_i \geq 0, \quad i = 1, 2, ..., p-re \tag{3.10}$$

Jelöljük $\mathbf{A}$-val és $\mathbf{b}$-vel azt a mátrixot illetve vektort, amelyek definiálják ezt a P poliédert. P nem üres, mert $\mathbf{c}/(\mathbf{y}_q \mathbf{c})$ vektor benne van.
Tudjuk, hogy létezik olyan $\mathbf{c}^* = (c_1^*, c_2^*, ..., c_p^*) \in P$, ami kielégíti az $\mathbf{A}'\mathbf{c}^* = \mathbf{b}'$ egyenletet, ahol $\mathbf{A}'$ az $\mathbf{A}$ mátrix $p \times p$-s nemszinguláris részmátrixa, illetve $\mathbf{b}'$ a $\mathbf{b}$ vektor p hosszú részvektora.
A Cramer-szabály szerint:

$$\mathbf{c}_i^* = \frac{det \mathbf{A}_i'}{det \mathbf{A}'},$$

ahol $\mathbf{A}'_i$-t úgy kapom meg $\mathbf{A}'$-ből, hogy az i. oszlopát a $\mathbf{b}$ vektorral helyettesítem.
Egy $m \times m$-es $\mathbf{M}$ mátrix determinánsa a következő alakban áll elő (jelöljük az i. sor j. elemét m_{ij}-vel):

$$det\,\mathbf{M} = \sum_{\sigma} sign(\sigma) \prod_{1 \le i \le p} m_{i,\sigma(i)},$$

ahol σ végigmegy az $\{1, 2, ..., p\}$ indexek permutációin, és $sign(\sigma) \in \{-1, 1\}$.
Ezért $|det\mathbf{M}| \le m^p \cdot p!$, ahol $m = \max |m_{ij}|$, ahol $i, j = 1, 2, ..., m$.
Az $\mathbf{A}$ mátrix, és ezért minden $\mathbf{A}'_i$ mátrix, továbbá a $\mathbf{b}$ vektor elemei a $[-3, 3]$ intervallumba eső egészek. Ezért $\left|det\mathbf{A}'_i\right| \le 3^p \cdot p!$ minden $i = 1, 2, ..., p$-re. Mivel $\left|det\mathbf{A}'\right| \ge 1$, ezért azt is tudjuk, hogy $\mathbf{c}^*_i \le 3^p \cdot p!$ minden $i = 1, 2, ..., p$-re, és

$$\mathbf{y}_j\mathbf{c}^* \le \sum \mathbf{c}^*_i \le p \cdot 3^p \cdot p!, \quad minden \quad j = 1, 2, ...q - ra.$$

Végül

$$1 = \mathbf{y}_q\mathbf{c}^* \le \frac{1}{2^{q-1}}\mathbf{y}_1\mathbf{c}^* \le \frac{1}{2^{q-1}} p \cdot 3^p \cdot p!,$$

ezért $q \le \log(p \cdot 3^p \cdot p!) + 1 = O(p \cdot \log p)$. □

Most már tudjuk bizonyítani az erősen polinomiális becslést az iterációk számáról.

3.2.17. Tétel. *A Newton metódus $O(p^2 \log p)$ iteráció alatt megtalálja egy lineáris kombinatorikus törtfüggvény optimalizálási feladat megoldását.*

Bizonyítás. Tekintsük a következő sorozatot:

$$S_i = H_{2i}B_{2i-1}.$$

A (3.5) egyenlőtlenség, és H_i, B_i monoton csökkenése (lásd (3.2.3) lemma) miatt:

$$S_{i+1} = H_{2i+2}B2i + 1 \le H_{2i+1}B2i + 1 \le \frac{1}{4}H_{21}B_{2i} \le \frac{1}{4}H_{21}B_{2i-1} = \frac{1}{4}S_i.$$

Másrészről

$$S_i = H_{2i}B2i - 11 = (A_{2i} - \delta B_{2i})B_{2i-1} = (A_{2i} - \frac{A_{2i-1}}{B_{2i-1}}B_{2i}) = A_{2i}B_{2i-1} - A_{2i-1}B_{2i}.$$

Azaz, vannak olyan $\mathbf{c} = (c_{11}, c_{12}, ..., c_{1p}, c_{21}, ...c_{pp}) \in \mathbb{R}^{p^2}$, és $\mathbf{y}_i \in \{0, 1, -1\}^{p^2}$ vektorok, amikre $S_i = \mathbf{c}\mathbf{y}_i$ minden i esetén, ahol

$$c_{ij} \in \{\alpha \cdot \beta : \alpha \in \{\mathbf{ax}, \mathbf{x} \in \mathcal{X}\}, \beta \in \{\mathbf{bx}, \mathbf{x} \in \mathcal{X}\}\}.$$

Így a (3.2.16) lemmát $\mathbf{c}$-re alkalmazva kapjuk, hogy az (S_i) sorozat legfeljebb $O(p^2 \log p^2) = O(p^2 \log p)$ hosszú, amiből következik a tételbeli becslés. □

3.2.18. Megjegyzés. Természetesen vannak más becslések is az iterációk számára. Az előző tételben ismertetett becslés a jelenleg ismert legjobb a lineáris kombinatorikus törtfüggvény optimalizációs feladatok Newton metódussal való megoldására. Ennek ellenére, ha egy konkrét esetet vizsgálunk, gyakran előfordul, hogy a feladat speciális tulajdonságai miatt jobb becsléseket kapunk.

3.2.19. Megjegyzés. Az erősen polinomiális becslés a lineáris kombinatorikus törtfüggvény optimalizálásra valahol meglepő, mivel a $h(\delta)$ függvény akár exponenciálisan sok lineáris részből is állhat.

3.2.20. Példa. Legyen az $\mathbf{a} = (a_1, a_2, ..., a_{3p})$ költségfüggvény, és a $\mathbf{b} = (b_1, b_2, ..., b_{3p})$ súlyfüggvény olyan, hogy

$$a_i = \begin{cases} 2^{i-1} & \text{i=1, ..., p} \\ 0 & \text{i=p+1, ..., 3p,} \end{cases}$$

$$b_i = \begin{cases} 0 & \text{i=1, ..., p} \\ 2^{i-p-1} & \text{i=p+1, ..., 3p.} \end{cases}$$

Legyen $\mathcal{U} \subseteq \{0,1\}^{3p}$ olyan halmaz, hogy az $\mathbf{x} = (x_1, x_2, ..., x_{3p}) \in \mathcal{U}$, ha $x_i = 0$ minden $i = p+1, ..., 3p$ indexre, illetve $\mathcal{W} \subseteq \{0,1\}^{3p}$ olyan halmaz, hogy az $\mathbf{x} = (x_1, x_2, ..., x_{3p}) \in \mathcal{W}$, ha $x_i = 0$ minden $i = 1, ..., p$ indexre. Az $\mathbf{au}$ skaláris szorzat 2^p féle értéket vehet fel, ha $\mathbf{u} \in \mathcal{U}$, mégpedig a $\{0, 1, ..., 2^p - 1\}$ értékeket, míg a $\mathbf{bw}$ skaláris szorzat 2^{2p} féle értéket vehet fel, ha $\mathbf{w} \in \mathcal{W}$, azaz $\mathbf{bw} \in \left\{0, 1, ..., 2^{2p} - 1\right\}$.
Minden $\mathbf{u} \in \mathcal{U}$ vektorra jelölje $\mathbf{w}_u$ azt a vektort, amelyre $\mathbf{bw}_u = (\mathbf{au})^2$. Végül legyen $\mathcal{X} = \{(\mathbf{u}, \mathbf{w}_u) : \mathbf{u} \in \mathcal{U}, \mathbf{u} \neq 0\}$.
Ha megvizsgáljuk azt a lineáris kombinatorikus törtfüggvény optimalizálási feladatot, amelynek inputja az $\mathcal{X}$ halmaz, és az $\mathbf{a}$, $\mathbf{b}$ vektorok, láthatjuk, hogy a h függvény a következőképpen néz ki:

$$h(\delta) = max\left\{k - \delta k^2 : k = 1, 2, ..., 2^p - 1\right\},$$

tehát $2^p - 1$ lineáris részből áll.

3.3. Megiddo algoritmusa a lineáris kombinatorikus törtfüggvény optimalizációra

Egy másik módszer a lineáris kombinatorikus törtfüggvény optimalizációs feladatok megoldására Megiddo paraméteres kereső algoritmusa, ami a következőképpen működik: egy $\mathcal{A}$ algoritmust futtatunk a lineáris alapprobléma megoldására, ami kiszámítja $h(\delta^*)$-t. Az alapötlet az, hogy

ugyan nem tudjuk előre a δ^* értékét, de szimbolikusan el tudjuk végezni a számolásokat, és így a δ^* értéke végül ki fog derülni.

3.3.1. Definíció. *Az $\mathcal{A}$ algoritmust lineárisnak nevezzük, ha minden összehasonlítás és számolás, amit az algoritmusban elvégzünk, az inputban szereplő számok két lineáris függvényét hasonlítja össze.*

3.3.2. Megjegyzés. Itt a lineáris algoritmus elnevezés tehát nem azt jelenti, hogy az algoritmus futásideje lineáris.

Ezért ha egy lineáris $\mathcal{A}$ algoritmusban össze kell hasonlítanunk egy x és y számot, akkor valójában $x(\delta^*)$-t kell összehasonlítanunk $y(\delta^*)$-gal, ahol $x(\delta)$, és $y(\delta)$ lineáris függvényei δ-nak. Nem kell tudnunk δ^* értékét, hogy elvégezzünk egy ilyen összehasonlítást. Elég meghatározni δ^* helyét az $x(\delta)$ és $y(\delta)$ függvények metszéspontjához, $\bar{\delta}$-hoz képest. Ez a probléma $h(\bar{\delta})$ előjelének kiszámolásával ekvivalens, amit a lineáris alapprobléma egy esetének megoldásával kapunk meg. Az algoritmus befejezéseképpen összegyűjtjük az információkat δ^*-ról, és így a lineáris alapprobléma egy újabb esetének megoldásával képesek leszünk kiszámítani δ^* pontos értékét.

Ha az $\mathcal{A}$ algoritmus futásideje T, akkor összesen $O(T)$ lineáris alapproblémát kell megoldanunk Megiddo algoritmusa közben, ezért a lineáris alapprobléma bonyolultságától függ az összes futási idő. Ez a lineáris részfeladatok számára vonatkozó $O(T)$-s becslés csökkenthető az $\mathcal{A}$ algoritmus párhuzamosságát vizsgálva. Megiddo [14] számos lineáris kombinatorikus törtfüggvény optimalizációs problémára alkalmazta a paraméteres keresést és hatékony algoritmusokat kapott azokban az esetekben, amikor volt hatékony párhuzamos algoritmus a lineáris alapproblémára.

Fontos megjegyezni, hogy a bináris kereséshez hasonlóan itt is elég, ha $\mathcal{A}_{dec}$-ra van eljárásunk, míg a Newton-metódushoz az "erősebb" változat, $\mathcal{A}_{opt}$ megoldása szükséges. Viszont itt az algoritmusunknak lineárisnak kell lennie, ami szintén erős feltétel.

A $\mathcal{A}_{opt}$ feladathoz kapcsolódva, definiáljuk a következő problémát:

$$\mathcal{P}_0(\delta): \quad \text{keressünk olyan } \mathbf{y} \in \mathcal{X} - et,$$

$$\text{amelyre } sign(\mathbf{ay} - \delta\mathbf{by}) = sign(\max\{\mathbf{ax} - \delta\mathbf{bx} : \mathbf{x} \in \mathcal{X}\}) = sign(h(\delta)).$$

3.3.3. Tétel. *Ha az $\mathcal{F}$ kombinatorikus törtfüggvény optimalizációs feladathoz tartozó $\mathcal{P}_0(\delta)$-ra létezik olyan $\mathcal{A}$ lineáris algoritmus, aminek T a futásideje, akkor az $\mathcal{F}$ feladat $O(T^2)$ időben megoldható.*

Bizonyítás. Futtassuk az $\mathcal{A}$ algoritmust a $\mathcal{P}_0(\delta^*)$ feladat megoldására, az ismeretlen δ^* paraméterrel. Mivel $\mathcal{A}$ lineáris algoritmus, ezért minden összehasonlítás az algoritmus során valójában egy ilyen egyenlőtlenséget vizsgál: $s - \delta^* t > 0$?, azaz $s/t > \delta^*$?, valamilyen ismert s, t

számokra. Egy ilyen összehasonlítást meg lehet oldani a $\mathcal{P}_0(s/t)$ probléma megoldásával, az $\mathcal{A}$ algoritmus futtatásával, $\delta = s/t$ paraméterrel, ugyanis az s/t, és δ^* közti kapcsolatot meg lehet állapítani $h(s/t)$ előjeléből. Ezért az összes számítás lefut $O(T^2)$ időben. $\square$

Valójában tehát $\mathcal{A}$-ból csinálunk egy új algoritmust a következő módosításokat elvégezve. Az algoritmusban szereplő minden c valós számot átírunk (c_1, c_2) alakba, ahol $c = c_1 - \delta^* c_2$. Így például az ismeretlen δ^* a $(0, -1)$ számpárnak fog megfelelni. Mivel tudjuk, hogy az $\mathcal{A}$ algoritmus lineáris, ezért minden összehasonlítás és számolás a δ^* lineáris függvénye, tehát valójában a (c_1, c_2), (d_1, d_2) alakú számokkal fogunk dolgozni. Az összeadásokat gond nélkül el tudjuk végezni: $(c_1, c_2)+(d_1, d_2) = (c_1+d_1, d_1+d_2)$, a szorzást azonban csak akkor engedjük meg, ha c_2 vagy $d_2 = 0$. Ekkor $(c_1, c_2) \cdot (d_1, d_2) = (c_1 d_1, c_1 d_2)$, ha $c_2 = 0$ volt, illetve $(c_1, c_2) \cdot (d_1, d_2) = (c_1 d_1, d_1 c_2)$, ha $c_1 = 0$ volt. Az összehasonlítások pedig így fognak kinézni az algoritmus futása során: $(c_1, c_2) > (d_1, d_2)$? Azaz $c_1 - \delta^* c_2 > d_1 - \delta^* d_2$? Átrendezve a következőt kapjuk: $(c_1 - c_2, d_1 - d_2) > 0$? Egy ilyen összehasonlítást azonban meg tudunk oldani a $\mathcal{P}_0(\frac{c_1 - d_1}{c_2 - d_2})$ feladat megoldásával, amihez csak futtatnunk kell az $\mathcal{A}$-t, mint szubrutint.

Ha kihasználjuk az $\mathcal{A}$ algoritmus összehasonlításainak lehetséges függetlenségét, jobb becsléseket kaphatunk, mint az előző tételben. Legyen $\mathcal{Q}$ egy olyan probléma, amelynek inputja tartalmazza a $\delta \in \mathbb{R}$ paramétert. A $\mathcal{Q}$ probléma megoldására készült algoritmus egy szintje a számítások egy része, amelyre igaz, hogy a szinten belüli bármely összehasonlítást a szint elejére mozdítva, az összehasonlítás eredménye nem változik. Csak olyan összehasonlításokat tekintünk, amelyeknek az eredménye csak a δ paraméter értékétől függ.

3.3.4. Definíció. *Egy algoritmus r szintből áll, ha az algoritmus számításait r fázisra tudjuk osztani, az aktuális input értékektől függetlenül.*

A következő tételben külön tekintjük az $\mathcal{A}_1$ főalgoritmust, amely az egész számításssorozatot vezérli, lineáris, valamint optimális esetben kevés szintre osztható, és az alárendelt $\mathcal{A}_2$ algoritmust, amelyet arra használunk, hogy megoldjuk az $\mathcal{A}_1$ algoritmus egyes számításait. Természetesen az $\mathcal{A}_1$ és $\mathcal{A}_2$ algoritmusok lehetnek ugyanazok, de ha a lehető legjobb futási időt szeretnénk elérni, általában nem egyeznek meg.

3.3.5. Tétel. *Ha van*

1. *$\mathcal{A}_1$ lineáris algoritmus egy $\mathcal{F}$ kombinatorikus törtfüggvény optimalizációs feladathoz tartozó $\mathcal{P}_0$ probléma megoldására, ami T_1 időben fut, r szintből áll, és az i. szint q_i számítást tartalmaz, és*

2. *$\mathcal{A}_2$ algoritmus a $\mathcal{P}_0$ probléma megoldására, ami T_2 időben fut,*

akkor az $\mathcal{F}$ probléma megoldható $O(T_1 + T_2 \sum_{1 \le i \le r} \log q_i)$ időben.

Bizonyítás. Futtassuk az $\mathcal{A}_1$ algoritmust a $\mathcal{P}_0$ probléma megoldására, ismeretlen δ^*-gal. Tekintsük a számítások i. szintjét. Ezen a szinten q_i független összehasonlítást kell elvégeznünk, amelyek $s - \delta^* t$ alakúak. Ahelyett, hogy elvégezzük egyesével az összes számítást, elég ha meghatározzuk δ^* és a q_i darab szám, azaz $x_1, x_2, ..., x_{q_i}$ kapcsolatát. Ha rendezzük ezeket a számokat, és δ^*-ot is elhelyezzük közöttük a megfelelő helyen, a bináris keresést használva, akkor ennek a szintnek a futási ideje $O(q_i \log q_i + T_2 \log q_i)$, hiszen az $\mathcal{A}_2$ algoritmust használjuk minden összehasonlításnál. Ebből következik, hogy az az egész probléma megoldására $O(T_1 max\left\{\log q_i\right\} + T_2 \sum_{1 \le i \le r} \log q_i)$ a futási idő.
Hogy megkapjuk a tételbeli futási időt, figyeljük meg, hogy az $x_1, x_2, ..., x_{q_i}$ számok rendezése elkerülhető. Ehelyett megkereshetjük a számok mediánját $O(q_i)$ időben, és elég, ha ezt a mediánt hasonlítjuk össze δ^*-gal az $\mathcal{A}_2$ algoritmust használva. Ennek az összehasonlításnak az eredménye megmondja nekünk a kapcsolatot δ^* és a számok fele között. Ezt ismételve, ismét megkeressük a maradék számok felének mediánját, és azt összehasonlítjuk δ^*-gal. Így végül a számokat két részre tudjuk osztani: a δ^*-nál nagyobbakra, és a δ^*-nál kisebbekre. A futási ideje

$$O(\sum_{0 \le k \le \log q_i} (\frac{q_i}{2^k} + T_2)) = O(q_i + T_2 \log q_i).$$

Összegezve ezt az összes szintre, a tételbeli futási időt kapjuk. □

Hogy minimalizálni tudjuk az előző tételbeli futási időt egy adott kombinatorikus törtfüggvény optimalizációs feladatra, az $\mathcal{A}_2$ algoritmusnak a lehető leggyorsabbnak kell lennie a probléma nem-tört alakú verziójára. Hogy az $\mathcal{A}_1$ algoritmusra is megtaláljuk a lehető legjobbat, érdemes megvizsgálni a párhuzamos algoritmusokat, mivel ezek kifejezetten úgy lettek kitalálva, hogy kevés szintjük legyen. Egy párhuzamos algoritmus, ami P processzort használva T ideig fut, úgy is tekinthető, mint egy nem párhuzamos, ami TP időben fut, és T szintje van, minden szinten legfeljebb P számítással. Ezért az alábbi tétel közvetlen következménye a (3.3.5) tételnek.

3.3.6. Tétel. *Ha van*

1. *$\mathcal{A}_1$ lineáris algoritmus egy $\mathcal{F}$ kombinatorikus törtfüggvény optimalizációs feladathoz tartozó $\mathcal{P}_0$ probléma megoldására, ami T_1 időben fut, és P processzort használ, és*
2. *$\mathcal{A}_2$ algoritmus a $\mathcal{P}_0$ probléma megoldására, ami T_2 időben fut,*

akkor az $\mathcal{F}$ probléma megoldható $O(T_1 P + T_2 T_1 \log P)$ időben.

Hogy még jobb futási időt kaphassunk, vizsgáljuk meg a következő feladatot, ugyanis az $\mathcal{A}_1$ algoritmusban gyakran előfordulhatnak ilyen alakú részfeladatok:

$$u(\delta^*) \leftarrow max\left\{u_1(\delta^*), u_2(\delta^*), ..., u_q(\delta^*)\right\}, \tag{3.11}$$

ahol u_i lineáris függvénye a δ paraméternek. Több megközelítését is bemutatjuk ennek a problémának, és próbálunk minél jobb becslést adni rá.

Ha ezt a (3.11) maximumot úgy számoljuk ki, hogy minden u_i-t ($i = 2, 3..., q$) összehasonlítunk az eddig már megtalált maximummal, akkor $O(T_2 q)$ idő szükséges a számításhoz, hiszen $q-1$ szintre tudjuk osztani, minden szinten egy összehasonlítással. Mint korábban is, T_2 jelöli az $\mathcal{A}_2$ algoritmus futási idejét, amit az $\mathcal{A}_1$ algoritmus összehasonlításainak megoldására használunk.

3.3.7. Állítás. *A (3.11) maximum $O(q + T_2 \log^2 q)$ időben kiszámítható.*

Bizonyítás. Ha az összehasonlítandó elemeket tournament fába rendezzük, $\lceil \log q \rceil$ szintet kapunk, minden szinten $O(q)$ összehasonlítással. Ekkor a (3.3.5) tétel miatt $O(q+T_2 \log^2 q)$ időben kiszámítható a (3.11) maximum. □

3.3.8. Állítás. *A (3.11) maximum $O(q + T_2 \log q \log\log q)$ időben kiszámítható.*

Bizonyítás. A szintek száma lecsökkenthető $O(\log\log q)$-ra a következőképpen: kövessük a tournament fát $O(\log\log\log q)$ szintig, hogy a még versenyben lévő elemek (amelyek eddig minden összehasonlításukat megnyerték) száma $k_1 \leq q/(\log\log q)$-ra csökkenjen. A továbbiakban az i. iterációban a k_i még versenyben lévő elemet osszuk $k_i^2/(2k_1)$ egyenlőre részre, és minden csoporton belül végezzük el az elemek összehasonlítását, hogy megtaláljuk a csoport legnagyobb elemét. Így az i. iterációban $2k_1 = O(q \ \log\log q)$ szükséges, és a még versenyben lévő elemek számát $k_{i+1} = k_i^2/(2k_1)$-re csökkenti. Tehát $k_i = k_1/2^{2^{i-1}-1}$, ezért $O(\log\log q)$ ilyen iterációt kell elvégeznünk. Összességében tehát ez az algoritmus q elem maximumának kiszámítására $O(q)$ összehasonlítást végez el $O(\log\log q)$ szinten. Ha ezt használjuk, mint $\mathcal{A}_1$ algoritmus, akkor szintén a (3.3.5) tétel felhasználásával $O(q + T_2 \log q \log\log q)$ futási időt kapunk a (3.11) maximum kiszámítására. □

3.3.9. Állítás. *A (3.11) maximum $O(q \log q + T_2 \log q)$ időben kiszámítható.*

Bizonyítás. A következőképpen is kiszámíthatjuk ezt a (3.11) maximumot: q lineáris függvény maximuma konvex, szakaszonként lineáris, és legfeljebb q töréspontja van. Ha van két konvex, szakaszonként lineáris függvényünk π_1 és π_2, egyenként q_1, illetve q_2 törésponttal, és ezek a törés pontok növekvő sorrendbe vannak rendezve az x koordinátájuk szerint, akkor a $max\{\pi_1, \pi_2\}$ függvény növekvő sorrendbe rendezett töréspontjait $O(q_1 + q_2)$ időben kiszámíthatjuk, egyszerűen merge rendezéssel. Így az $u_1, u_2, ..., u_q$ lineáris függvények maximumának töréspontjait $O(q \log q)$ időben meghatározhatjuk. Azt a lineáris részt megtalálni pedig, amelyik tartalmazza q^*-ot, a bináris kereséssel tudjuk, a töréspontokat használva. Így összességében a (3.11) maximum $O(q \log q + T_2 \log q)$ időben számítható ki. □

3.3.10. Definíció. *Egy $\mathcal{A}$ algoritmus maximumszámítási fázisa az algoritmus egy része, amelyben minden összehasonlítás részt vesz a maximumok kiszámításban, és ezek a maximumok függetlenek, ezért tetszőleges sorrendben elvégezhetőek a számítások, anélkül, hogy megváltozna az eredményük.*

A következő tételben az utóbbi két (3.11) maximumszámítási módszert fogjuk felhasználni. Legyen q elem maximumának a mérete q.

3.3.11. Tétel. *Ha van*

1. *$\mathcal{A}_1$ lineáris algoritmus egy $\mathcal{F}$ kombinatorikus törtfüggvény optimalizációs feladathoz tartozó $\mathcal{P}_0$ probléma megoldására, ami T_1 időben fut, r maximumszámítási fázisa van, és a maximumok teljes mérete az i. fázisban legfeljebb q_i, és*

2. *$\mathcal{A}_2$ algoritmus a $\mathcal{P}_0$ probléma megoldására, ami T_2 időben fut,*

akkor az $\mathcal{F}$ probléma megoldható
(i) $O(T_1 + T_2 \sum_{1\le i\le r} \log q_i \log\log q_i)$ időben, és
(ii) $O(T_1 + \sum_{1\le i\le r}(q_i + T_2)\log q_i)$ időben.

Bizonyítás. Futtassuk az $\mathcal{A}_1$ algoritmust a $\mathcal{P}_0$ probléma megoldására az ismeretlen δ^* paraméterrel, és tekintsük az i. fázisát a számításoknak. Ekkor legfeljebb q_i méretű, egymástól független maximumokat kell kiszámítanunk. Ha ezeket párhuzamosan számítjuk az $O(q \log\log q)$ szintes, korábban említett algoritmussal, akkor ennek a fázisnak a futási ideje $O(q_i + T_2 \log q_i \log\log q_i)$, amiből megkapjuk a tételbeli (i) teljes futási időt.
Ha viszont először kiszámítjuk az összes maximum töréspontját, rendezzük az összes töréspontot, és elhelyezzük δ^*-ot a rendezett listában a bináris kereséssel, akkor az i. fázis futási ideje $O((q_i + T_2)\log q_i)$, és ezt összegezve kapjuk az egész algoritmus futási idejére a tételbeli (ii) becslést. □

4. fejezet

Hiperbolikus programozási feladatok

4.0.12. Definíció. *Hiperbolikus programozási feladatnak nevezünk egy optimalizálási feladatot, ha lineáris feltételrendszer nemnegatív megoldásait tartalmazó halmaz felett olyan racionális törtfüggvény optimumát keressük, amelyben mind a számláló, mind a nevező lineáris elsőfokú függvény.*

A hiperbolikus elnevezés abból a tényből adódik, hogy egyváltozós esetben az $f(x) = \frac{cx+c_0}{dx+d_0}$ célfüggvény képe hiperbola. Szokás még lineáris programozási feladatnak nevezni törtlineáris célfüggvénnyel.

Az általános lineáris programozási feladatoknál egy adott célfüggvényhez kerestük az optimális megoldást. Előfordulhat azonban, hogy több szempont együttes figyelembevételével szeretnénk optimális megoldást találni. Ha például az egyik szempont szerint minimalizálni, a másik szerint maximalizálni szeretnénk, akkor a két célfüggvény hányadosának optimumát keresve olyan megoldást találunk, amely a lehető legkisebb első érték mellett a lehető legnagyobb a második érték szerint.

Műszaki és gazdasági területeken gyakran előfordulnak olyan optimalizálási modellek, ahol valamilyen hatékonysági mutatószámot (pl. termelékenység, egységnyi ráfordításra jutó teljesítmény, egységnyi nyereségre jutó ráfordítás stb.) akarunk optimalizálni. Hasonlóképpen, ha egy bizonyos mennyiség átlagát akarjuk optimalizálni, akkor is ilyen hiperbolikus programozási feladathoz jutunk.

A hiperbolikus programozási feladat általános alakja a következő:

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{x} &\geq 0 \\ max \quad & \frac{\mathbf{cx} + c_0}{\mathbf{dx} + d_0}, \end{aligned} \tag{4.1}$$

ahol $\mathbf{A} \in \mathbb{R}^{n\times p}$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{c}, \mathbf{d} \in \mathbb{R}^p$, $c_0, d_0 \in \mathbb{R}$ adottak, és $\mathbf{x} \in \mathbb{R}^p$ a programozási feladat döntési változója. Jelölje $P = \{\mathbf{x} \in \mathbb{R}^p : \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq 0\}$ a lehetséges megoldások halmazát. Tegyük fel, hogy a célfüggvény nevezője nem 0, azaz $\mathbf{dx} + d_0 \neq 0$ a P feltételi halmaz pontjaiban.

4.0.13. Megjegyzés. Ha $\mathbf{d} = 0$ és $d_0 = 1$, akkor egy lineáris programozási feladatot kapunk. A c_0 szám ekkor konstans a célfüggvényben, amely az optimalizálás szempontjából érdektelen.

4.0.14. Megjegyzés. Valójában elég lenne a max $\frac{cx}{dx}$ alakú optimalizálási feladatot megoldani, hiszen az általános alakú feladat célfüggvénye segédváltozók bevezetésével ilyen alakra hozható.

4.1. A célfüggvény vizsgálata

A P feltételi halmazról tudjuk, hogy konvex poliéder, sőt a továbbiakban azt is feltesszük, hogy korlátos. Ekkor a tört célfüggvény felveszi optimumát (maximumát és minimumát is) a P konvex politop valamely extremális pontjában. Ezen kívül feltesszük, hogy a célfüggvény nevezője a P feltételi halmazon nem nulla. Ebből a feltevésből a következő állítás szerint több is következik.

4.1.1. Állítás. *Legyen $S \subseteq \mathbb{R}^p$ konvex halmaz. Ha $\boldsymbol{dx} + d_0 \neq 0$ minden $\boldsymbol{x} \in S$ esetén, akkor a $\boldsymbol{dx} + d_0$ lineáris függvény az S halmaz minden pontjában azonos előjelű, azaz $\boldsymbol{dx} + d_0 > 0$, vagy $\boldsymbol{dx} + d_0 < 0$ minden $\boldsymbol{x} \in S$ vektorra.*

Bizonyítás. Indirekt tegyük fel, hogy van olyan $\mathbf{x}_1, \mathbf{x}_2 \in S$, hogy $\mathbf{dx}_1 + d_0 > 0$ és $\mathbf{dx}_2 + d_0 < 0$. Ekkor az $\mathbf{x}_1$ és $\mathbf{x}_2$ pontokat összekötő szakasznak biztosan van olyan $\mathbf{x}_3$ pontja, ahol $dx_3 + d_0 = 0$, mivel a lineáris függvény folytonos. Ekkor ellentmondásra jutottunk, mert azt is tudjuk, hogy $\mathbf{x}_3 \in S$, mivel S konvex. □

Mint láttuk, az a kikötés, hogy a célfüggvény nevezője ne legyen nulla azt jelenti, hogy a célfüggvény értéke pozitív vagy negatív értékű minden lehetséges megoldás esetén. A továbbiakban mindig feltesszük, hogy a nevező pozitív a megoldási halmazon. Amennyiben negatív lenne, a célfüggvény számlálóját és nevezőjét is (-1)-el szorozva a feltételezett esetet kapjuk.

Az az előbbi állítás, miszerint a megoldások korlátos konvex poliéderén a tört célfüggvény felveszi optimumát, a következő állításból következik. A bizonyítását nem közöljük, de a [12] cikkben megtalálható.

4.1.2. Állítás. *Legyen $S \subseteq \mathbb{R}^p$ konvex halmaz. Legyen az $f(\boldsymbol{x}) = \frac{\boldsymbol{cx}+c_0}{\boldsymbol{dx}+d_0}$ függvény olyan, hogy $\boldsymbol{dx} + d_0 \neq 0$ minden $\boldsymbol{x} \in S$ vektorra. Ekkor az $f(\boldsymbol{x})$ lineáris törtfüggvény szigorúan kvázikonvex, és szigorúan kvázikonkáv is az S halmazon.*

Hivatkozva a kvázikonvex függvények konvex poliédereken történő optimalizálására, azon belül is arra tételre, mely szerint egy folytonos kvázikonvex függvény felveszi maximumát a

kompakt poliéder valamely extremális pontjában, megkapjuk, hogy a hiperbolikus programozási feladatokban a célfüggvény felveszi optimumát, ha a korábban közölt feltevésekkel élünk.

Kérdés, hogyan tudjuk eldönteni egy konkrét feladatban, hogy a feltételek teljesülnek-e, tehát, hogy a P feltételi halmaz korlátos, és a célfüggvény nevezője nem nulla P-n.

A nevező vizsgálatát úgy szoktuk elvégezni, hogy a nevezőre vonatkozó minimum és maximum feladatot külön-külön megoldjuk, azaz az alábbi lineáris programozási feladatokat kell megoldani:

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{x} &\geq 0 \\ min \quad & \mathbf{dx} + d_0 \\ max \quad & \mathbf{dx} + d_0 \end{aligned}$$

Ha mindkét érték pozitív, akkor a nevező pozitív előjelű a feltételi halmaz minden pontjában, ha mindkét érték negatív, akkor mindenhol negatív.

A feltételi halmaz korlátosságának vizsgálatát a következő lineáris programozási feladat megoldásával lehet eldönteni:

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{x} &\geq 0 \\ max \quad & x_1 + x_2 + ... + x_p \end{aligned}$$

Ha a célfüggvény maximális értéke véges, akkor a P feltételi halmaz korlátos.

4.1.3. Megjegyzés. Nagyon sok esetben egyszerűen el lehet dönteni a lineáris feltételekből, hogy a kikötések teljesednek-e.

4.2. Visszavezetés lineáris programozási feladatra

A következőkben azt a megoldási módszert ismertetjük, amely a hiperbolikus programozási feladat megoldását liníaris programozási feladat megoldására vezeti vissza. Amerikai módszernek, vagy Charnes-Cooper módszernek is nevezik két kifejlesztője után. A lineáris programozási feladatot egy új változó és egy új feltétel bevezetésével kapjuk meg.

Szorozzuk be a feltételeket, és a célfüggvény számlálóját és nevezőjét is egy pozitív t is-

meretlennel. Az alábbi feladatot kapjuk:

$$\begin{aligned} \mathbf{A}\mathbf{x}t - \mathbf{b}t &= 0 \\ \mathbf{x} &\geq 0 \\ t &> 0 \\ max \quad & \frac{\mathbf{c}\mathbf{x}t + c_0 t}{\mathbf{d}\mathbf{x}t + d_0 t} \end{aligned}$$

A korábbiak alapján feltehetjük, hogy $\mathbf{d}\mathbf{x}t + d_0 t > 0$. Az új ismeretlen értéke legyen olyan, hogy az új nevező értéke legyen minden x megoldásra 1, azaz $\mathbf{d}\mathbf{x}t + d_0 t = 1$. Ekkor az új célfüggvény értéke a számláló értékével lesz egyenlő. Ezzel elértük, hogy az új feladatban nem szerepel nevező, igaz a feltételek és ismeretlenek száma eggyel nőtt.

Az új feladatban változók szorzatai is szerepelnek, így még nem lineáris a feladat. Könnyen belátható, hogy az $\mathbf{y} = \mathbf{x}t$ új változó bevezetésével már lineáris feladatot nyerünk, amely az alábbi:

$$\begin{aligned} \mathbf{A}\mathbf{y} - \mathbf{b}t &= 0 \\ \mathbf{x} &\geq 0 \\ t &\geq 0 \\ \mathbf{d}\mathbf{y} + d_0 t &= 1 \\ max \quad & \mathbf{c}\mathbf{y} + c_0 t \end{aligned} \tag{4.2}$$

Ugyan itt a formalitás kedvéért a t változó nemnegativitását tettük fel, de vegyük észre, hogy a nevező pozitivitása miatt t értéke is csak pozitív lehet.

Úgy is tekinthetjük ezt az átalakítást, hogy bevezetjük a $t = \frac{1}{\mathbf{d}\mathbf{x}+d_0}$ új változót. Itt is feltesszük, hogy $\mathbf{d}\mathbf{x} + d_0 > 0$. Legyen $\mathbf{y} = \mathbf{x}t$, és így is az előbb leírt lineáris programozási feladatot kapjuk.

Az (4.1) és (4.2) feladatok ekvivalensek a következő értelemben: ha x megoldja a (4.1) feladatot, akkor $t = \frac{1}{\mathbf{d}\mathbf{x}+d_0} > 0$, és $y = tx$ megoldja a (4.2) feladatot, azaz van olyan (y, t) megoldása, amelyre $t > 0$. Fordítva, ha y és t együtt megoldja a (4.2) feladatot, akkor $t > 0$, és $x = \frac{1}{t}y$ megoldja a (4.1) feladatot.

4.3. Alkalmazásai

Az ütemezési feladatok vizsgálata az 50-es évek elején kezdődött, majd tekintettel a feladat gyakorlati fontosságára sok különböző modell tanulmányozására került sor, és a témakör nagyon gyors és nagy fejlődésen ment keresztül. A modellek száma igen nagy, már 1977-ben 9000-re becsülték a kategorizálható különböző determinisztikus ütemezési problémák számát. Azóta ez a

szám még számottevően növekedett. A továbbiakban bemutatunk egy nagyon egyszerű modellt, amelyet lineáris törtprogramozásra fogunk visszavezetni.

A matematikai modellt döntési problémaként vezetjük be, egy döntési alapszituációra építve. Először tehát ismertetjük a döntési alapproblémát.

Egy termelési folyamatban n-féle terméket állítanak elő m-féle erőforrás felhasználásával. A j. termék egységének előállításához az i. erőforrásból a_{ij} mennyiséget használnak el valamilyen egységben mérve. Tudjuk, hogy az egyes erőforrásokból mennyi áll rendelkezésre, jelölje az i. erőforrásból rendelkezésre álló mennyiséget b_i, és jelölje $x_1, ..., x_n$ az egyes termékekből gyártandó mennyiségeket, ezek lesznek a döntési változóink, a kiszámítandó értékek.
Foglaljuk össze az a_{ij} értékeket az $\mathbf{A}$ mátrixban ($\mathbf{A} \in \mathbb{R}^{n \times m}$), a b_i értékeket a $\mathbf{b}$ vektorban ($\mathbf{b} \in \mathbb{R}^m$), az $x_1, ..., x_n$ változókat az $\mathbf{x}$ vektorban ($\mathbf{x} \in \mathbb{R}^n$).
Feltesszük, hogy x_j mennyiség gyártásához az i. forrásból $a_{ij}x_j$ mennyiséget fogunk felhasználni, hogy az egyes forrásokból a felhasználások összeadódnak, így az i. forrásból az összes felhasználás $\sum_{1 \leq i \leq n} a_{ij}x_j$, és hogy az egyes termékekből gyártandó mennyiségeknek nem kell egészértékűeknek lenniük.

Ezek a feltevések konkrét feladat esetében nem biztos, hogy megállják a helyüket. Ha például az egyik erőforrás a munkaóra, természetes feltételezés lehet, hogy tört számú munkaórára nem alkalmazhatunk senkit, vagy hogy másik termék gyártására nem csoportosítható át a munkaerő gond nélkül, azaz a munkaórák nem adódnak össze. Az egyes termékekből rendszerint egész számú mennyiségeket gyártanak. Ez gyakran mégsem okoz problémát, ha elég kicsi az egység, amiben mérjük a szóban forgó terméket, máskor azonban nincs mód erre sem. Ha hajót vagy vonatot gyártunk például, akkor elő kell írnunk a gyártandó mennyiség egészértékűségét. Itt azonban csak olyan esetekkel foglalkozunk, amikor az ilyen problémák nem bírnak nagy jelentőséggel.

Ekkor azt a feltételt, hogy az egyes erőforrásokból legfeljebb annyit használunk fel, amennyi a rendelkezésünkre áll, és hogy nem gyártunk negatív mennyiséget, felírhatjuk lineáris feltételek formájában, vagyis így:

$$\mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{x} \geq 0.$$

A cél lehet többek között a termelés hatékonysága. A hatékonyság mérésére szokásos az egységnyi költségre eső árbevételt alkalmazni. Jelölje $d_1, d_2, ..., d_p$ az egyes termékek egy egységének előállítási költségét, $c_1, c_2, ..., c_p$ az eladásukból származó árbevételt. Az összes költség, és az összes árbevétel kiszámításánál élünk a linearitási feltétellel. Foglaljuk össze az adott d_i értékeket a $\mathbf{d}$ vektorban, és a c_i értékeket a $\mathbf{c}$ vektorban. Ekkor a termelés hatékonyságát, amelyet maximalizálni szeretnénk, a következő hányados fejezi ki:

$$\frac{\sum_{1 \leq i \leq p} c_i x_i}{\sum_{1 \leq i \leq p} d_i x_i} = \frac{\mathbf{cx}}{\mathbf{dx}}.$$

4.3.1. Példa. Egy üzemben két terméket gyártanak két erőforrás igénybevételével. A termelés során a termékek egységének megmunkálási költsége rendre 2, 5 pénzegység. A termékek szerelési költsége rendre 1, 2 pénzegység. A termékek eladási egységára várhatóan rendre 3, 9 pénzegység. A két termékből összesen legalább 60 darabot szeretnénk előállítani. Azt szeretnénk, hogy a szerelési költség összesen legfeljebb 80 pénzegység legyen. A termékmennyiségtől független költség 10 pénzegység, és ezt a megmunkálási költségnél vesszük figyelembe. Határozzuk meg azt a termékösszetételt, amelynél az egységnyi megmunkálási költségre eső árbevétel maximális!

Írjuk fel a feladat matematikai modelljét. Legyen a döntési változó az egyes termékek gyártandó mennyisége, jelölje ezeket x_1, x_2. Az alábbi hiperbolikus programozási feladatot kaptuk:

$$\begin{aligned} x_1 + x_2 &\geq 60 \\ x_1 + 2x_2 &\leq 80 \\ x_1, x_2 &\geq 0 \\ max \quad & \frac{3x_1 + 9x_2}{2x_1 + 5x_2 + 10} \end{aligned}$$

A nevező minden lehetséges értéke legalább 10, mert a változók nemnegatívok, így az első feltétel teljesül. A feltételi halmaz korlátossága is látszik, hiszen a második egyenlőtlenség miatt $0 \leq x_1 \leq 80$, és $0 \leq x_2 \leq 40$.

A formális beszorzások, és az $y_i = x_i t$, $(i = 1, 2)$ helyettesítés után a következő lineáris programozási feladatot kapjuk:

$$\begin{aligned} 2y_1 + 5y_2 + 10t &= 1 \\ y_1 + y_2 - 60t &\geq 0 \\ y_1 + 2y_2 - 80t &\leq 0 \\ y_1, y_2, t &\geq 0 \\ max \quad & 3y_1 + 9y_2 \end{aligned}$$

Ezt pedig szimplex algoritmussal megoldva azt kapjuk, hogy az optimális megoldás:

$$y_1 = \frac{40}{190}, \quad y_2 = \frac{20}{190}, \quad t = \frac{1}{190}, \quad z_{max} = \frac{300}{190}.$$

A hiperbolikus programozási feladat optimális megoldása tehát ($x_i = \frac{z_i}{t}$, $i = 1, 2$):

$$x_1 = 40, \quad x_2 = 20, \quad z_{max} = \frac{300}{190}.$$

5. fejezet

Alkalmazások

5.1. Maximális súlyozott átlagköltségű utak aciklikus gráfokban

Példaként a lineáris törtfüggvény optimalizálási problémákra tekintsük elsőként a maximális súlyozott átlagköltségű utakat aciklikus gráfokban. A probléma egy esetének inputja tartalmaz egy $n \geq 1$ egész számot, egy n pontú aciklikus gráf éleinek E halmazát, és az élek egy $\mathbf{a} : E \to \mathbb{R}$ költségfüggvényét, illetve $\mathbf{b} : E \to \mathbb{R}$ súlyfüggvényét. Az egyszerűség kedvéért feltesszük, hogy nincsenek többszörös élek, és a csúcsok topológikus sorrendbe vannak rendezve, azaz, ha $(u, v) \in E$, akkor $u < v$, valamint, hogy minden csúcs elérhető az 1. csúcsból.
Ha P egy út, $f(P) = \sum_{(u,v)\in P} \mathbf{a}(u, v)$ az út költsége, és $g(P) = \sum_{(u,v)\in P} \mathbf{b}(u, v)$ az út súlya. A feladat:

$$MaxRatioP: \quad \frac{f(P)}{g(P)}, \text{ ahol P út az 1. csúcsból az n. csúcsba.}$$

Itt az $\mathcal{X}$ struktúrák halmaza az összes $1 - n$ utat reprezentáló $x \in \{0, 1\}^{|E|}$ vektort tartalmazza. A problémához tartozó paraméteres probléma a következőképpen néz ki:

$$MaxP(\delta): \quad max\left\{f(P) - \delta g(P)\right\}, \text{ ahol P út az 1. csúcsból az n. csúcsba.}$$

Ha δ, és a P út rögzített, akkor $f(P) - \delta g(P) = \sum_{(u,v)\in P}(\mathbf{a}(u, v) - \delta\mathbf{b}(u, v))$, azaz a $MaxP(\delta)$ probléma optimális megoldása, a maximális költségű 1-n út, az $\mathbf{a} - \delta\mathbf{b}$ költségfüggvény szerint.

Ismert, hogy aciklikus gráfokban legrövidebb, illetve leghosszabb utat $O(m)$ időben tudunk keresni a topológikus sorrendbe rendezett csúcsokat, és a belőlük kijövő éleket végigvizsgálva. Minden csúcsra definiáljuk a $d[v]$ címkét, ami az aktuális leghosszabb v-be vezető út költségét tartalmazza, és ezt frissítjük, ha találunk egy jobbat. Legyen az algoritmus neve $MaxCost$. Így minden $\delta \in \mathbb{R}$ számra a $MaxP(\delta)$ probléma $O(m)$ időben megoldható. A (3.2.17) miatt tehát a $MaxRationP$ feladat $O(m^2 \log m)$ iteráció alatt, azaz összességében $O(m^3 \log p)$ idő alatt megoldható. Azonban a feladat speciális tulajdonságait felhasználva jobb becslést is kaphatunk

az iterációk számára, ha az élsúlyok mind nemnegatívak. Ekkor valójában elég $O(m \log n)$ iteráció (lásd [3]), azaz $O(m^2 \log n)$ idő.

Megiddo paraméteres keresését használva, a (3.3.3) tétel miatt $O(m^2)$ idő szükséges a megoldáshoz, ugyanis a maximális, azaz δ^* költségű $1-n$ út az $\mathbf{a} - \delta^*\mathbf{b}$ költségfüggvény szerint 0 költségű. Ezért Megiddo algoritmusában az aktuális v csúcsból u csúcsba induló élhez tartozó összehasonlítás így néz ki: $d[v] + a(v,u) - \delta^* b(v,u) > d[u]$? Ez valójában egy $s - \delta^* t > 0$? alakú összehasonlítás, ahol az s és t számokat ismerjük. Hogy megtudjuk egy ilyen összehasonlítás eredményét, valójában meg kell tudnunk a kapcsolatot s/t és δ^* között, amit a $MaxP(s/t)$ probléma előjelének kiszámításával tudunk megmondani.

Használjuk ki a leghosszabb utat kiszámító algoritmus párhuzamosságát, azaz hogy az adott csúcsból kijövő éleket egyszerre is vizsgálhatjuk, anélkül, hogy megváltozna az összehasonlítások eredménye. Valójában az ismeretlen δ^*-ot, és s_k/t_k $(k = 1, ..., deg(v))$ számokat kell összehasonlítanunk, amit az egyesével való összehasonlítás helyett (minden alkalommal a $MaxCost$ algoritmust kell futtatnunk) megtehetünk úgy, hogy rendezzük az s_k/t_k számokat, és beillesztjük δ^*-ot a megfelelő helyre bináris keresés alkalmazásával. Így $O(\log(deg(v)))$ alkalommal kell futtatnunk a $MaxCost$ algoritmust, a bináris keresés minden lépéséhez. Ha ehhez hozzáadjuk $deg(v)$ szám rendezését, összességében $O(deg(v) \log deg(v)) + O(m \log deg(v)) = O(m \log deg(v))$ idő szükséges a v csúcs feldolgozásához. Ezért az összes futási idő:

$$O(m \sum_{1 \le v \le n-1} \log(deg(v))) = O(mn \log(m/n)).$$

Még ennél is kaphatunk jobb futási időt, ha a (3.3.11) tételt használjuk. Azonban ehhez egy másik algoritmus kell a leghosszabb út kiszámítására. Ez a rekurzív algoritmus külön tekinti a gráf első $k = \lfloor (n+1)/2 \rfloor$ pontja által feszített részgráfot, majd azokat az (u,v) éleket, amelyekre $1 \le u \le k \le v \le n$, és végül az utolsó $n-k$ csúcs által feszített részgráfot vizsgálja meg. Ugyanúgy $O(m)$ időben fut, mint a $MaxCost$ algoritmus, de az a speciális tulajdonsága, hogy maximum számítási fázisokra bontható. Ezért alkalmazhatjuk rá a (3.3.11) tétel (ii) részét, így $O(mn)$-es futási időt kapunk. (Bővebben lásd: [3].)

5.2. Maximális súlyozott átlagköltségű feszítőfa keresése

Tekintsünk egy másik példát a lineáris kombinatorikus törtfüggvény optimalizációs problémára. A maximális súlyozott átlagköltségű feszítőfa probléma egy esete tartalmaz egy összefüggő, irányítatlan $G(V,E)$ gráfot, egy $\mathbf{a} : E \to \mathbb{R}$ költségfüggvényt, és egy $\mathbf{b} : E \to \mathbb{R}$ súlyfüggvényt. A G gráf egy feszítőfájának $\mathbf{a}(T)$ költsége, illetve $\mathbf{b}(T)$ súlya a feszítőfa éleinek összköltsége, illetve

összsúlya. A $\mathcal{T}$ feladat:

$$\max \quad \frac{\mathbf{a}(T)}{\mathbf{b}(T)}, \quad \text{ahol } T \text{ a } G \text{ gráf feszítőfája.}$$

A $\mathcal{T}$ feladathoz tartozó $\mathcal{P}(\delta)$ paraméteres probléma megtalálni a maximális költségű feszítőfát az $\mathbf{a} - \delta\mathbf{b}$ költség szerint. Ismeretes, hogy a maximális költségű feszítőfa megtalálásának problémája, ami ekvivalens a minimális költségű feszítőfa megtalálásának problémájával, $T_{mf} = O(\min\{m \log\log n, m + n \log n\})$ időben megoldható (lásd [3]).

A maximális súlyozott átlagköltségű feszítőfa az $\mathbf{a} - \delta^*\mathbf{b}$ költségfüggvény szerinti maximális költségű feszítőfa. Mivel a maximális költségű feszítőfa csak az éleinek költség szerinti sorrendjén múlik, ezért hogy megtaláljuk az $\mathbf{a} - \delta^*\mathbf{b}$ költségfüggvény szerinti maximális költségű feszítőfát, elég ha rendezzük az $\mathbf{a}(e) - \delta^*\mathbf{b}(e)$ számokat, minden $e \in E$ élre. Ezt az m számot $O(\log m)$ időben tudjuk rendezni, ha m processzort használunk. Így a (3.3.6) tétel alapján a maximális súlyozott átlagköltségű feszítőfa keresésének problémája

$$O(m \log m + T_{mf} \log^2 m) = O(T_{mf} \log^2 m)$$

időben oldható meg.

5.3. Minimális súlyozott átlagkör keresése gráfokban

A minimális súlyozott átlagkör keresésének problémája újabb példa a lineáris kombinatorikus törtfüggvény optimalizációra. Legyen $G(V, A)$ egy irányított n pontú gráf, $\mathbf{a} : A \to \mathbb{R}$ és $\mathbf{b} : A \to \mathbb{R}$ az éleinek költségfüggvénye, illetve súlyfüggvénye. A $\mathcal{C}_{ir}$ feladat:

$$min \quad \frac{\mathbf{a}(C)}{\mathbf{b}(C)}, \quad \text{ahol } C \text{ a } G \text{ gráf köre.}$$

A korábbiakból tudjuk, hogy ha van algoritmusunk a probléma nem tört alakú verziójára, akkor azt mint szubrutint felhasználva meg tudjuk oldani az eredeti feladatot is. Itt azonban a természetesnek tűnő lehetőség, hogy keressünk minimális köröket, nem működik, mivel a minimális kör megkeresése **NP**-teljes probléma.

Nézzük meg, hogy működne ez az uniform esetben, amikor minden él súlya azonosan 1, tehát a feladat a $\delta^* = min\ \frac{\mathbf{a}(C)}{k}$ érték megtalálása, ahol k a gráf C körének élszáma.

Ekkor a minimális átlagkör keresésének paraméteres változata a következő: keressük azt a legkisebb δ^* költséget, amivel megváltoztatva $\mathbf{a}$-t, minden kör átlagköltsége nemnegatív lesz az új költségfüggvény szerint. Ugyanis, legyen C^* a minimális átlagkör, és $\frac{\mathbf{a}(C^*)}{k}$ a minimum értéke, és tetszőleges C körre az átlagköltség $\frac{\mathbf{a}(C)}{l}$, és tekintsük ennek a C körnek az emelés utáni új költségét: $\mathbf{a}(C) - l\frac{\mathbf{a}(C^*)}{k}$. Ekkor az emelés utáni új átlagköltsége:

$$\frac{\mathbf{a}(C) - l\frac{\mathbf{a}(C^*}{k}}{l} \geq 0,$$

ami azért igaz, mert $\frac{\mathbf{a}(C)}{l} - \frac{1}{k}\mathbf{a}(C^*) \geq 0$, azaz $\frac{\mathbf{a}(C)}{l} \geq \frac{\mathbf{a}(C^*)}{k}$, ami igaz, mert azt tettük fel, hogy C^* a minimális átlagú kör. Ezért, ha pont $\left|\frac{\mathbf{a}(C^*)}{k}\right|$-val növelem a költségeket minden élen, akkor az összes kör nemnegatív lesz.

Ezért a megoldási ötlet az, hogy keressünk egy minimális költségű C kört, aminek az átlagköltsége $\delta = \frac{\mathbf{a}(C)}{l}$, és emeljük meg minden él költségét $-\delta$-val. Ekkor C új átlagköltsége 0, és minden legalább l hosszú C' kör átlagköltsége nemnegatív lesz, ugyanis $\mathbf{a}(C')$-höz hozzáadunk egy nála nagyobb abszolútértékű számot. Ezért legfeljebb élszámnyi növeléssel az összes kör költsége nemnegatív, és az utolsóként tekintett kör költsége 0 lesz.

Sajnos azonban ez a megoldási módszer nem működik a korábban említett okból, hogy a minimális kör megkeresése **NP**-teljes probléma, sem a speciális uniform, sem az általános esetben. Az általános esethez visszatérve tudjuk azonban, hogy egy irányított gráfban a minimális súlyozott átlagú, legfeljebb n élű körséta értéke megegyezik a minimális súlyozott átlagú kör értékével, ezért az új feladat a minimális súyozott átlagú, legfeljebb n élű körséta megtalálása. Ennek a feladatnak a nem tört alakú verzióját, tehát a minimális költségű legfeljebb n élű körséta megtalálását viszont már könnyen meg tudjuk oldani. Keressük például rekurzívan a minimális költségű, legfeljebb k élű $s-s$ sétákat minden s-e, és minden $k = 1, 2, ..., n$ számra. Ezt $O(n^2m)$ időben tudjuk megoldani. A Newton algoritmus $O(m^2 \log m)$ iteráció alatt véget ér, és minden iterációban meg kell oldani a nem tört alakú szubrutint, tehát $O(n^2m^3 \log m)$ lépés szükséges a probléma megoldására.

Nézzük meg ugyanezt a feladatot nem irányított gráfok esetén is.

Legyen $G(V, E)$ egy irányítatlan n pontú gráf, $\mathbf{a} : E \to \mathbb{R}$ és $\mathbf{b} : E \to \mathbb{R}$ az éleinek költségfüggvénye, illetve súlyfüggvénye. A $\mathcal{C}_{nir}$ feladat:

$$min \quad \frac{\mathbf{a}(C)}{\mathbf{b}(C)}, \quad \text{ahol } C \text{ a } G \text{ gráf köre.}$$

Hogy meg tudjuk oldani a feladatot, itt is szükséges egy algoritmus a nem tört alakú probléma megoldására. Vegyük észre, hogy nem irányított gráfban az igaz, hogy a minimális átlagköltségű kör érétke megegyezik a minimális átlagköltségű Euler részgráf értékével (hiszen felbomlik élidegen körök uniójára, tehát valójában egy körséta, és ráadásul legfeljebb n éle van). Ezért ennek a feladatnak a nem tört alakú verzióját kell megoldanunk, ami tehát a legolcsóbb Euler részgráf keresése. Ez $O(n^4)$ időben megtehető (lásd [15]). Mivel a feladat megoldásakor a Newton metódus $O(m^2 \log m)$ iteráció alatt véget ér, és minden iterációban egyszer meg kell oldanunk a nem tört alakú részfeladatot, ezért az összes futási idő $O(n^4m^2 \log m)$.

Irodalomjegyzék

[1] Thomasz Radzik, *Parametric Flows, Weighted Means of Cuts, and Fractional Combinatorial Optimization*, Complexity in Numarical Optimization, 1993, 351-386 oldalak

[2] Thomasz Radzik, *Newton's Method for Fractional Combinatorial Optimization*, Standford University, 1992

[3] Du, D. Z., P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Kluwer Academic Publisher, Dordrecht-Boston-London, 1998, 430-472 oldalak

[4] Jüttner Alpár, *Paraméteres problémák a kombinatorikus optimalizálásban, és ezek távközlési alkalmazásai*, Doktori értekezés, 2006, 69-63 oldalak

[5] S. Schaible, *A survey of fractional programming*, Generalized Concavity in Optimization and Economics, Academic Press, New York, 1981, 417-440 oldalak

[6] S. Schaible, T. Ibaraki, *Fractional programming*, Europ. J. of Operational Research, 12, 1983

[7] W. Dinkelbach, *On nonlinear fractional programming*, Managment Science, 1967

[8] T. Ibaraki, *Parametric approaches to franctional programs*, Math. Programming, 1983

[9] P. M. Pardalos, A. T. Phillips, *Global optimization of fractional programs*, J. Global Opt., 1991

[10] A. V. Karzanov, *On minimal mean cuts and circuits in a digraph*, Mathods for Solving Operator Equations, 1985, 72-83 oldalak

[11] Imreh Balázs, Imreh Csanád, *Kombinatorikus optimalizálás*, Novadat, 2005

[12] Dr. Nagy Tamás, *Hiperbolikus optimalizálás*, Miskolci egyetem, Alk. Matematikai Tanszék

[13] Komáromi Éva, *Lineáris programozás*, Budapesti Közgazdaságtudományi és Államigazgatási Egyetem, Operációkutatás Tanszék, 2005, 51-53 oldalak

[14] N. Megiddo, *Combinatorial optimization with rational objective functions*, Math. of Oper. Res., 1979

[15] W. J. Cook, W. H. Cunningham, W.R. Pulleyblank, A. Schrijver *Combinatorial Optimization*,