Eötvös Loránd Tudományegyetem

Természettudományi Kar

Halasi Valentina

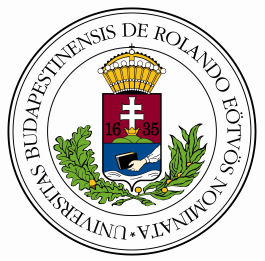
Polinomiális lineáris programozási módszerek

BSc Alkalmazott Matematikus Szakdolgozat

Témavezető:

Jüttner Alpár

Operációkutatási Tanszék



Budapest, 2016

Tartalomjegyzék

Bevezetés

A lineáris programozás egy lineáris megengedettségi problémája adott $A \in \mathbb{Z}^{m \times n}$ mátrix és $b \in \mathbb{Z}^m$ vektor esetén az alábbi rendszer megengedett megoldását megtalálni:

$$Ax = b \ , \ x \geq 0, \tag{1}$$

vagy bizonyítani, hogy az egyenletrendszernek nincs nemnegatív megoldása. Az első gyakorlatban használt algoritmus a szimplex módszer volt, egy kombinatorikus algoritmus, melyet Dantzig [1] publikált 1947-ben. Bár a gyakorlatban hatékony, futási ideje bizonyos feladatokra akár exponenciális is lehet. Később számos algoritmus született a feladat megoldására, mint Agmon, Motzkin és Schoenberg relaxációs módszere 1954-ben. Az első polinomiális futási idejű algoritmus az ellipszoid módszer volt, melynek polinomiális voltát Khachiyan [2] orosz matematikus igazolta 1978-ban. Neumann János Dantziggal való levelezésében ismertette az első belsőpontos módszert a lineáris megengedettségi feladat megoldására. Ez az algoritmus nem polinomiális idejű, és a gyakorlatban sem hatékony, ám számos, az utóbbi évtizedekben született polinomiális futási idejű algoritmus alapjául szolgál. Neumann algoritmusát Dantzig [6] publikálta 1984-ben. Ezt követte Karmarkar [7] belsőpontos módszere ugyan ebben az évben. Chubanov [4] 2012-ben egy, a relaxációs módszeren alapuló algoritmust tett közzé, illetve 2013-ban [11] egy Neumann [6] algoritmusán alapuló módszere is született, melyet ez a dolgozat is bemutat.

E szakdolgozat témája olyan, az utóbbi 15 évben született algoritmusok részletezése, melyekben egyszerű iteratív módszereket és újraskálázó lépéseket kombinálva a lineáris programozás polinomiális futási idejű algoritmusaihoz jutunk. Bár ezek a módszerek lassabbak, mint néhány belső pontos módszer, abból a szempontból előnyösebbek lehetnek, hogy a bennük használt legbonyolultabb műveletek vektor- és mátrixszorzások.

Az első fejezetben két fent említett algoritmust, Neumann algoritmusát és a relaxiációs módszert foglaljunk össze. Ezen kívül Dunagan és Vempala [8] 2008-ban publikált algoritmusának koordinátaváltoztató lépését is ismertetjük itt, hiszen ez a második és harmadik

fejezetekben tárgyalt két Dadush-Végh-Zambelli-algoritmus [6] alapja. A 4. fejezet témája a már említett Chubanov [11] algoritmus, végül Végh László és Giacomo Zambelli [5] relaxációs módszeren alapuló buborék algoritmusa zárja a dolgozatot. A függelékben a dolgozatban használt jelölések szerepelnek, illetve a kódolási méret fogalmának részletezése.

1. fejezet

Korábbi iteratív lineáris programozási algoritmusok

Ebben a fejezetben olyan algoritmusok rövid leírása található meg, melyek a későbbi fejezetekben ismertetett módszerek alapjául szolgálnak.

Számos algoritmus ismert az $\mathcal{L}_+ = \{x \in R^n_+ : Ax = 0\}$, az $\mathcal{A}_+ = \{x \in R^n_+ : Ax = b\}$ vagy az $\mathcal{L}^{\perp}_+ := \{z \in R^n_+ : \exists y \in R^n : z = Ay\}$ poliéderek egy nem nulla pontjának megtalálására. A legtöbb iteratív módszerben az alábbi lépések találhatóak meg $\mathcal{L}_+$ egy pontjának meghatározásához:
Minden iteráció során fenntartunk egy nemnegatív $x \in R^n$ vektort, ami nem a nullvektor. Az x vektorból kiszámoljuk y-t az $y = Ax$ képlettel, e vektor normájának csökkentése a cél.
(1) Ha $y = 0$, akkor x az L_+ egy pontja.
Ha $A^T y > 0$, akkor $A^T y \in L_> = \{x \in R^n_> : Ax = 0\}$
(2) Különben kiválasztunk egy olyan $k \in [n]$ indexet, melyre $a_k y \leq 0$ és újradefiniáljuk a használt vektorokat a következőképpen:

$$y' := \alpha y + \beta \hat{a}_k,\ x' := \alpha x + \frac{\beta}{\|a_k\|} e_k, \tag{1.1}$$

ahol az α és β szorzótényezők algoritmusonként változnak.
Ezen algoritmusok konvergenciájának vizsgálatánál egy fontos mennyiség a kondíciószám, melyet ρ-val jelölünk és az alábbi módon definiálható:

$$\rho_A := \max_{\|y\|=1, y \in \mathbb{R}^m} \min_{j \in [n]} \hat{a}_j^T y.$$

Geometriailag $|\rho_A|$ az origó távolsága a $\text{conv}(\hat{A})$ halmaz határától. $\rho_A > 0$ akkor és csakis akkor, ha az origó $\text{conv}(\hat{A})$-on kívül helyezkedik el. Ekkor ρ_A az $\{y \in \mathbb{R}^m : A^T y > 0\}$ duális kúpba írható legnagyobb gömb sugara, melynek középpontja az origó középpontú egységgömb egy határpontja. Illetve $\rho_A < 0$ pontosan akkor teljesül, ha az origó $\text{conv}(\hat{A})$ belsejében van, ekkor $-\rho_A$ a legnagyobb origó középpontú $\text{conv}(\hat{A})$-ba írt gömb sugara. Különben $\rho_A = 0$.

1.1. Dunagan és Vempala algoritmusa

Dunagan és Vempala [8] algoritmusában $\alpha = 1$ és $\beta = -(\hat{a}_k^T y)$, tehát

$$y' := y + -(\hat{a}_k^T y)\hat{a}_k,\ x' := x + \frac{-(\hat{a}_k^T y)}{\|a_k\|} e_k.$$

β ezen választása azért célszerű, mert $\alpha = 1$ esetén $\|y'\|$ ekkor lesz a lehető legkisebb. y normája így a következőképpen változik:

$$\|y'\| = \|y - (\hat{a}_k^T y)\hat{a}_k\| = \|y\|(y' - (\hat{a}_k^T \hat{y})\hat{a}_k) =$$

$$\|y\|\sqrt{\|\hat{y}\|^2 - 2\hat{y}\hat{a}_k^T\hat{y}\hat{a}_k + \hat{a}_k^T\hat{y}\hat{a}_k\hat{a}_k^T\hat{y}\hat{a}_k} = \|y\|\sqrt{1 - (\hat{a}_k^T\hat{y})^2}.$$

Egy iteráció során minél nagyobb az a_k és az y által bezárt szög, annál kisebb annak szinusza (az utolsó gyökjel alatti kifejezés éppen ez az érték). Tehát a legnagyobb normacsökkenést annak az oszlopvektornak a felhasználásával érhetjük el, amelyik a legnagyobb szöget zárja be az y vektorral.
Ha van olyan k index, amelyre $\hat{a}_k^T y \leq -\varepsilon$, akkor vektorváltoztatást hajtunk végre. y' normája ekkor jelentősen csökken: $\|y'\| \leq \|y\|\sqrt{1-\varepsilon^2}$

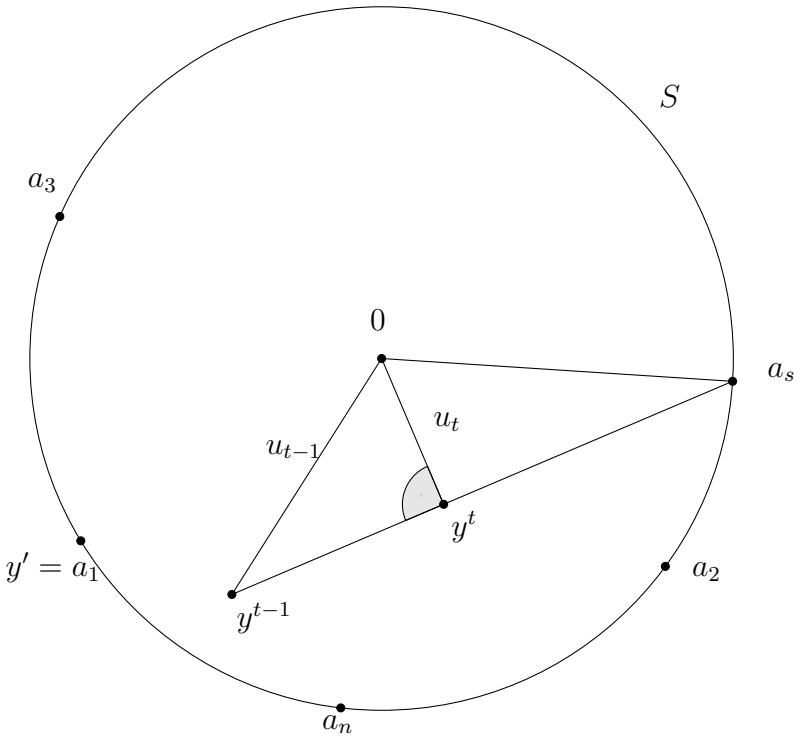
1.2. Neumann algoritmusa

Neumann algoritmusát egyszerűsége miatt gyakran választják új lineáris programozási algoritmusok alapjául, de a gyakorlatban ritkán alkalmazzák lassú konvergenciája miatt. Itt az y vektor az A oszlopvektorainak konvex kombinációja. Az α, $\beta > 0$ paramétereket úgy választjuk, hogy $\alpha + \beta = 1$ és $\|y'\|$ a lehető legkisebb, tehát y' a legkisebb normájú pont az y-t a_k-val összekötő szakaszon. Az alábbi rendszer egy megengedett megoldását

keresi Neumann algoritmusa:

$$\begin{aligned} Ax &= 0, \\ \mathbf{1}x &= 1, \\ x &\geq 0, \end{aligned}$$

ahol $A \in \mathbb{R}^{m \times n}$ és $\|a_j\| = 1 \; \forall j \in [n]$.



1.1. ábra. Neumann algoritmusának egy lépése: y^t meghatározása

Geometriailag az A mátrix a_j oszlopai az S origó középpontú, 1 sugarú gömbfelszínen fekvő pontok. A feladat olyan nemnegatív x_j súlyokat találni a a_j pontokhoz, hogy súlyozott középpontjuk a 0 legyen.
Az algoritmus az origó bármely x^0 közelítő vektorával inicializálható, melyre $x^1 \geq 0$ és $\mathbf{1}x^1 = 1$. Az első iteráció során

$$x_1^1 \coloneqq 1,\; x_j^1 \coloneqq 0 \;\forall\; j \neq 1,\; y^1 = a_1,\; u_1 \coloneqq \|y_1\|,\; t \coloneqq 2.$$

A t. iteráció leírása ($t \geq 2$): Adott egy közelítő megoldás:

$$x = x^{t-1} \geq 0,\ \sum_1^n x_j^{t-1} = 1,\ y^{t-1} = \sum a_j x_j^{t-1},\ u_{t-1} = \|y^{t-1}\|.$$

(1) Keressük meg azt az a_j pontot, melyre az $(a_j, 0, y^{t-1})$ szög a legnagyobb:

$$s \coloneqq \operatorname*{argmin}_j (y^{t-1})^T a_j;$$

(2) Legyen $v = (y^{t-1})^T a_s$. Ha $v > 0$, akkor minden a_j pont a 0-na átmenő, az y^{t-1} pont irányára merőleges hipersík által meghatározott egyik féltéren helyezkedik el. Így a lineáris megengedettségi feladat nem megoldható, mivel a a_j pontoknak nem lehet olyan konvex kombinációja, mely a 0-t eredményezi.
(3) Válasszuk ki a következő y^T közelítő pontot, mely az origóhoz legközelebbi pont a y^{t-1}-et a_s-el összekötő szakaszon.

$$\begin{aligned} y^t &\coloneqq \lambda y^{t-1} + (1-\lambda) a_s; \\ u_t^2 &\coloneqq \lambda v + (1-\lambda); \\ x_j^t &\coloneqq \lambda x_j^{t-1} \text{ ha } j \neq s \text{ és } x_s^t \coloneqq \lambda x_s^{t-1} + (1+\lambda); \end{aligned}$$

ahol λ-t úgy választjuk, hogy $u_t = \|y^t\|$ minimális legyen:

$$\lambda \coloneqq (1-v)/(u_{t-1}^2 - 2v + 1).$$

λ-ra teljesül, hogy $0 < \lambda < 1$, mivel $v = (y^{t-1})^T a_s \leq 0$. Illetve $u_t < u_{t-1}$, mivel u_{t-1} az $(y^{t-1}, y^t, 0)$ derékszögű háromszög átfogója, $u_t = 0y^t$ pedig ugyanezen háromszög egyik befogója.
(4) $t \coloneqq t+1$, go to (1)

Neumann algoritmusa minden iteráció során egyszerű számításokat hajt végre. A legkomplikáltabb művelet a mátrix-vektor szorzás a 2. lépésben, amikor az a_s oszlopot kiválasztjuk, ez $O(nm)$ műveletet igényel. Az algoritmus konvergenciáját Dantzig [9], illetve Epelman és Freund [10] vizsgálták. Az utóbbi szerzőpáros megállapítása alapján ha az origó az a_i vektorok konvex burkában van, akkor $\|y'\| \leq \|y\|\sqrt{1-\rho_A^2}$ minden iteráció során.

1.3. Relaxációs módszer

Adott az $Ax \leq b$ egyenlőtlenségrendszer, a x^0 kezdőpont és a $\lambda > 0$ szám. Határozzuk megy a $x^1, x^2, \ldots,$ pontokat a következőképpen.
Ha az i iterációban létrehozott a x^i vektorra $Ax^i \leq b$, egy megoldást találtunk, és az algoritmus megáll.
Különben létezik egy olyan $a_k x \leq b_k$ egyenlőtlenség, melyet x^i megsért. Legyen ekkor x^{i+1} a következő:

$$x^{i+1} \coloneqq x^i + \lambda \frac{(b_k - a_k x^i)}{\|a_k\|^2} \cdot a_k^T.$$

x^{i+1}-et tehát a x^i pontnak az $a_k x = b_k$ egyenlet megoldásai által meghatározott hipersíkra való merőleges vetítésével kapjuk. Ha $\lambda = 2$, akkor x^{i+1} a z^i tükörképe erre a hipersíkra, $\lambda = 1$ esetén pedig a merőleges vetülete. Motzkin és Schoenberg megmutatták, hogy $\lambda = 2$ választással esetén az algoritmus befejeződik, ha az $\{x : Ax \leq b\}$ teljes dimenziójú.

2. fejezet

Dadush, Végh és Zambelli algoritmusa Dunagan-Vempala skálázással

Legyen $A = [a_1, a_2, \ldots, a_n]$ egy $m \times n$-es m-rangú egész számokból álló mátrix. A fejezetben ismertetett algoritmus [6] az $\{x \in \mathbb{R}^n, Ax = 0\}$ poliéder egy minden koordinátájában pozitív x pontját keresi koordinátacsökkentő iterációkkal és újraskálázó lépésekkel. Az algoritmus kibővíthető az $Ax = 0$, $x \geq 0$ rendszer maximális tartójú megoldásának keresésének problémájára.
Az

$$Ax = 0 \ , \ x > 0 \tag{2.1}$$

primál feladat az $Ax = 0$, $-x \leq -1$ rendszer megoldhatóságával ekvivalens, így a dualitás tételből következik, hogy a primál poliéder akkor és csakis akkor nem üres, ha az

$$yA \geq 0 \ , \ (yA)\mathbf{1} > 0 \tag{2.2}$$

duális feladatnak nincs megoldása.

Az algoritmus a Dunagan és Vempala által használt koordinátaváltoztató lépést alkalmazza az A mátrix oszlopain. Addig hajtja végre ezeket az iterációkat, amíg $\|y\|$ lényeges csökkenését tapasztaljuk, különben újraskálázzuk a mátrixot, hogy geometriailag átalakítsuk a problémát.

2.1. Az algoritmus leírása

Az alábbiakban ismertetett algoritmus a következő paramétereket használja:

$$\varepsilon := \frac{1}{11m} \ , \ N := 6mL \ , \ \delta := \min_{j \in 1 \ldots n} \frac{1}{\|(AA^T)^{-1} a_j\|} \tag{2.3}$$

Algorithm 1

Input: Egy $A \in \mathbb{Z}^{m \times n}$ m rangú mátrix

Output: Az $Ax = 0$, $x > 0$ rendszer megoldása, vagy tanúsítvány arra, hogy nem megoldható

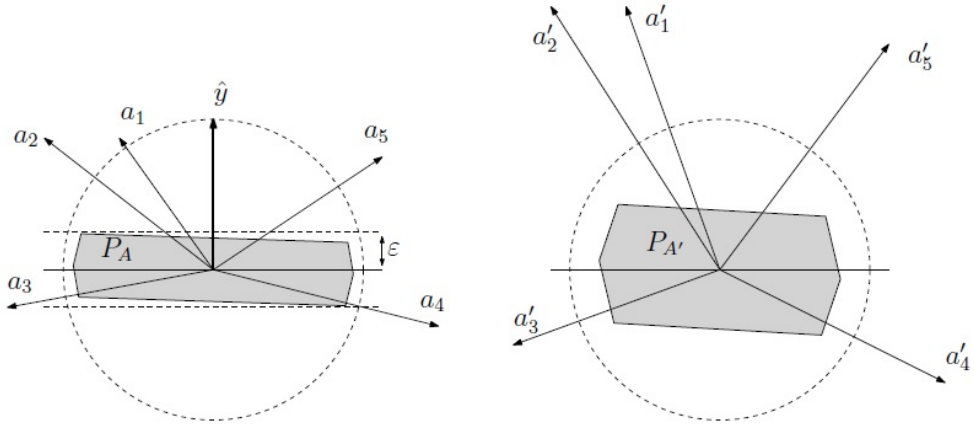
Legyen $x_j := 1 \ \forall j \in n$ és $y := Ax$. Legyen $t := 0$.
while $\|y\| \geq \delta$ és $t \leq N$ **do**
  if $A^T y \geq 0$ **then**
    STOP, az $\{x \in \mathbb{R}^n, Ax = 0, x > 0\}$ poliéder üres
  else legyen $k := arg\min_{j \in n} \hat{a}_j^T \hat{y}$
    if $\hat{a}_k^T y < -\varepsilon$ **then** koordinátaváltoztatás
      $y := y - (\hat{a}_k^T y)\hat{a}_k$; $x := x - \frac{\hat{a}_k^T y}{\|a_k\|^2} e_k$
    else újraskálázás:
      $A := (I + \hat{y}\hat{y}^T)A$;
      $y := 2y$;
      $t := t + 1$;
    end if
  end if
end while
if $\|y\| < \delta$ **then**
  a rendszer megoldása $x := x - A^T(AA^T)^{-1}y$;
else nincs megoldás
end if

Az újraskálázással a célunk a kondíciószám javítása. $|\rho_A|$ helyett a P_A politóppal dolgozunk, melyet a következőképpen definiálunk:

$$P_A := conv(\hat{A}) \cap (-conv(\hat{A})). \tag{2.4}$$

$|\rho_A|$ a legnagyobb P_A-ba írt origó középpontú gömb sugara.
Ha $\hat{a}_j^T \hat{y} \geq -\varepsilon$, akkor $P_A \subseteq \{z \in \mathbb{R}^n : -\varepsilon \leq \hat{y}^T z \leq \varepsilon\}$ (a 2.1 ábrán ezt a szürke alakzat

szemlélteti). Ha az A mátrixot az $A' \coloneqq (I + \hat{y}\hat{y}^T)A$ mátrixra cseréljük, megmutatható, hogy $P_{A'}$ térfogata legalább 3/2-szerese P_A térfogatának. Geometriailag A' egy olyan lineáris transzformációval kapható, mely A oszlopait $\hat{y}$ irányába kétszeresére nyújtja.
Tehát minden iteráció során az aktuális y vektor hosszának lényeges csökkenését, vagy P_A térfogatának konstans szorzóval való növekedését érjük el. Mivel P_A térfogata legfeljebb akkora, mint az m dimenziós egységgömb térfogata, így az algoritmus nem hajthat végre túl sok átskálázást, ha az eredeti P_A poliéder nem üres.
Az A mátrix méretében polinomiális számú iteráció után arra jutunk, hogy (1) nem megoldható, vagy kapunk egy y vektort, melynek euklideszi normája δ -nál kisebb. Ez utóbbi esetben megmutatjuk, hogy az aktuális x vektor merőleges vetülete A nullterére a lineáris megengedettségi feladat egy megoldása.



2.1. ábra. Egy újraskálázó lépés, mely P_A-t $\hat{y}$ irányában (az ábra forrása: [6])

2.2. Térfogatváltozás

Az alábbi lemma a P_A poliéder egy átskálázó lépés utáni térfogatváltozásáról szól, mely fontos az algoritmus polinomiális voltának igazolásához.

2.2.1. Lemma. *Tegyük fel, hogy* (1) *megoldható. Legyen* $0 < \varepsilon < 1/(11m)$, $v \in \mathbb{R}^m$, $\|v\| = 1$ *úgy, hogy* $\hat{a}_j^T v \geq -\varepsilon \ \forall j \in 1 \ldots n$. *Legyen* $A' = (I + vv^T)A$.
Ekkor $vol(P_{A'}) \geq \frac{3}{2} vol(P_A)$.

Bizonyítás. *Legyen $A \coloneqq (I + vv^T)$. Elegendő megmutatnunk, hogy $TP_A \subseteq (1+3\varepsilon)P_{A'}$. Ebből már következik a lemma, mivel $vol(TP_A) = 2vol(P_A)$, felhasználva, hogy bázistranszformációval kiszámolható, hogy $det(T) = 2$.*

Ebből ara jutunk, hogy $vol(P_A) \geq 2vol(P_A)/(1+3\varepsilon)^m \geq \frac{3}{2}vol(P_A)$, ahol az utolsó egyenlőtlenség az alábbiból következik: $ln(1+3\varepsilon)^m \leq ln(1+3/(11m))^m \leq \frac{3}{11} \leq ln\frac{4}{3}$.

$TP_A \subseteq (1+3\varepsilon)P_{A'}$ bizonyításához tekintsünk egy tetszőleges $z \in P_A$ pontot. P_A szimetrikussága miatt elegendő belátni, hogy $Tz \in (1+3\varepsilon)conv(\hat{A}')$. Definíció szerint létezik $\lambda \in \mathbb{R}^n_+$ vektor, hogy $\sum_{j=1}^n \lambda_j = 1$ és $z = \sum_{j=1}^n \lambda_j \hat{a}_j$. Ekkor

$$Tz = \sum_{j=1}^{n} \lambda_j T\hat{a}_j = \sum_{j=1}^{n} (\lambda_j \|T\hat{a}'_j\| \hat{a}'_j) = \sum_{j=1}^{n} \lambda_j \sqrt{1 + 3(v^T\hat{a}_j)^2}\hat{a}'_j \tag{2.5}$$

Mivel feltettük, hogy $0 \in conv(\hat{A}')$, így elegendő belátni a következő egyenlőtlenséget:

$$\sum_{j=1}^{n} \lambda_j \sqrt{1 + 3(v^T\hat{a}_j)^2} \leq 1 + 3\varepsilon.$$

Az egyenlőtlenség bal oldalán lévő tagra tekinthetünk úgy, mint a $v^T\hat{a}_j$ értékeket λ_j valószínűséggel felvevő valószínűségi változó várható egy függvényének várható értékére. Tudjuk, hogy $v^T\hat{a}_j$ a $[-\varepsilon, 1]$ intervallumon veszi fel értékeit és $|E[X]| = |\sum_{j=1}^n \lambda_j v^T\hat{a}_j| = |v^Tz|$.

Tekintsük az $X \in \mathbb{R}$ valószínűségi változót, mely a $[-\varepsilon, \eta]$ intervallumon veszi fel az értékeit, ahol $0 \leq \varepsilon \leq \eta$ és $E[X] = \mu$. A lemma bizonyításához elegendő megmutatni, hogy $E[\sqrt{1+cX^2}] \leq \sqrt{1 + c\eta(\varepsilon + \mu)}$. Ezt alkalmazva $X = v^T\hat{a}_j$ és $\eta = 1$ helyettesítésével ugyanis

$$\sum_{j=1}^{n} \lambda_j \sqrt{1 + 3(v^T\hat{a}_j)^2} \leq 1 + 3(\varepsilon + |v^Tz|) \leq 1 + 3(2\varepsilon) \leq 1 + 3\varepsilon.$$

A második egyenlőtlenségnél kihasználtuk, hogy z és $-z$ is a P_A poliéderben van, így $|v^Tz| \leq \varepsilon$.

A várható érték függvényének felsőbecsléséhez tekintsük az

$$l(x) = \frac{\eta - x}{\eta + \varepsilon}\sqrt{1 + c\varepsilon^2} + \frac{x + \varepsilon}{\eta + \varepsilon}\sqrt{1 + c\eta^2}$$

affin interpolációs függvényét $\sqrt{1+cx^2}$-nek a $[-\varepsilon, \eta]$ intervallumon. Mivel $\sqrt{1+cx^2}$ egy konvex függvény, így $l(x) \geq \sqrt{1+cx^2}$ minden $x \in [-\varepsilon, \eta]$ pontban.

Ezáltal $E[\sqrt{1+cX^2}] \leq E[l(X)] = l(E[X]) = l(\mu)$, mivel l egy affin függvény.

Ebből azt kapjuk, hogy

$$l(\mu) = \frac{\eta-\mu}{\eta+\varepsilon}\sqrt{1+c\varepsilon^2} + \frac{\mu+\varepsilon}{\eta+\varepsilon}\sqrt{1+c\eta^2} \leq \sqrt{1+c\left(\frac{\eta-\mu}{\eta+\varepsilon}\varepsilon^2 + \frac{\mu+\varepsilon}{\eta+\varepsilon}\eta^2\right)} =$$

(mivel $\sqrt{x}$ konkáv függvény)

$$= \sqrt{1+c(\eta\varepsilon+(\eta-\varepsilon)\mu)} \leq \sqrt{1+c\eta(\varepsilon+|\mu|)},$$

ahol az utolsó egyenlőtenségnél kihasználtuk, hogy $\varepsilon \leq \eta$. □

2.3. Paraméterbecslések

Az algoritmus futási idejének megállapításához néhány az A mátrix kódolási méretétől függő paramétert szükségeltetik becsülni. Az alábbi két állítás bizonyításánál felhasználjuk, hogy egy A mátrix minden B négyzetes részmátrixára teljesül, hogy $|det(B)| \leq 2^L$ (5.6.1).

2.3.1. Állítás. $\delta \geq 2^{-3L}$.

Bizonyítás. Egy B mátrix sorainak illetve oszlopindexeinek részhalmazát P-vel és Q-val jelölve legyen $B_{P,Q}$ azon részmátrixa B-nek, melynek sorai a P halmazban lévő indexeknek megfelelő sorok és oszlopai a Q halmazban lévő indexeknek megfelelő oszlopok. Legyen $M \coloneqq AA^T$. A Cauchy-Binet formula szerint $\forall \leq k \leq m$ és bármely $P, Q \subseteq [m]$, $|P| = |Q| = k$ választással

$$det(M_{P,Q})^2 = \sum_{\substack{U\subseteq[m]\\|U|=k}} det(A_{P,U})det(A_{Q,U}) \leq \binom{n}{k}2^{2L} \leq 2^{3L},$$

ahol az utolsó egyenlőtlenségnél kihasználjuk, hogy $\binom{n}{k} \leq n^k$ és $L \geq m \cdot logn$.
Emiatt az M mátrix adjungált mátrixának, $adj(M)$-nek minden tagjának abszolútértéke legfeljebb $2^{\frac{3}{2}L}$, így az $adj(M)a_j$ vektor minden koordinátája legfeljebb $m2^{\frac{5}{2}L}$, $j \in [n]$. Következésképpen $\|adj(M)a_j\|^2 \leq m^3 2^{5L} \leq 2^{6L}$. Mivel $M^{-1} = \frac{adjM}{detM}$, így $\delta \geq \frac{detM}{2^{3L}} \geq 2^{-3L}$.
□

2.3.2. Állítás. *Ha az origó $conv(A)$ belsejében van, akkor $conv(A) \supseteq B(0, 2^{-2L})$ és $\|\rho_A\| \geq 2^{-3L}$.*

Bizonyítás. Az állítás első felének bizonyításához legyen p egy az origóhoz legközelebbi pont $conv(A)$ határán. Mivel p a $conv(A)$ egy oldala tartalmazza, így létezik egy $m \times m$-es nemszinguláris B részmátrixa A-nak úgy, hogy p az origó merőleges vetülete a $H \coloneqq \{y \in \mathbb{R}^m : \exists x$ úgy, hogy $y = Bx, e^T x = 1\}$ halmazra. Ha $(B^{-1})^T e$ -t γ -val jelöljük, akkor $H = \{y \in \mathbb{R}^m : \gamma^T y = 1\}$, így $p = -\gamma/\|\gamma\|^{-1}$. Mivel a B mátrix minden elemének abszolútértéke legfeljebb 2^L, így γ minden koordinátája legfeljebb $\frac{m2^L}{det(B)}$. Így $\|\gamma\| \leq \frac{m^{3/2}2^L}{det(B)} \leq 2^L$, ahol az utolsó egyenlőtlenség a $2^L \geq L \geq mn \geq m^{3/3}$ becslésből és abból következik, hogy $det(B) \geq 1$, mivel A egész számokból álló mátrix.
Az állítás második részének igazolásához legyen $\alpha = \min_{j\in[n]} \|a_j\|$. Ekkor $conv(\hat{A}) \subseteq \alpha^{-1} conv(A)$, így $|\rho_A| \geq \alpha^{-1}\|p\|$. Mivel $\alpha \leq 2^L$ kapjuk, hogy $|\rho_A| \geq (\alpha\|\gamma\|)^{-1} \geq 2^{-3L}$. $\square$

2.4. Az algoritmus helyessége

2.4.1. Tétel. *Egy adott $A \in \mathbb{Z}^{m\times n}$ mátrixra az algoritmus az az $Ax = 0$, $x > 0$ lineáris megengedettségi probléma egy x megoldását adja akkor és csakis akkor, ha az megoldható. A while ciklus iterációinak száma $O(m^3 L)$, a teljes műveletigény pedig $O((m^3 n + mn^2)L)$.*

A probléma nem megoldható, ha az algoritmus talál egy olyan $y \neq 0$ vektort, melyre $A^T y \leq 0$, hiszen y tanúsítvány a duál feladat megoldhatóságára (mivel $\|y\| > \delta$, így $y \neq 0$). Tekintsük azt az esetet, amikor az algoritmus N újraskálázás után megáll, és indirekt tegyük fel, hogy (1) megoldható. Ekkor $conv(A)$ tartalmazná az origót, így a 2.3.2. Állítás miatt az eredeti P_A poliéder tartalmazna egy legalább 2^{-3L} sugarú gömböt. Tehát $vol(P_A) \geq V_m 2^{-3mL}$ lenne kezdetben, ahol V_m az m-dimenziós egységgömb térfogatát jelöli. A 2.2.1 Lemma miatt N iteráció után $vol(P_A) \geq (3/2)^N 2^{-3mL} V_m > V_m$, ami ellentmondáshoz vezet, mivel P_A az m-dimenziós egységgömbben van.
Tekintsük azt az esetet, amikor az algoritmus $\|y\| \leq \delta$ feltétellel megáll, jelöljük a két output vektort $\bar{x}$-al és $\bar{y}$-al. Vegyük észre, hogy minden iteráció során fenntartottuk, hogy $y = Ax$ és $x_j \geq 1$ $\forall j \in [n]$. Tekintsük az A kezdeti mátrixot és legyen $\bar{A}$ az algoritmus utolsó iterációjában használt alapmátrix, tehát $\bar{y} = \bar{A}\bar{x}$. Legyen x' az algoritmus által szolgáltatott megoldás, vagyis $x' = \bar{x} - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{y}$. Ekkor $\bar{A}x' = \bar{A}(\bar{x} - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{y}) = \bar{A}\bar{x} - \bar{y} = 0$ (x' az $\bar{x}$ merőleges vetülete az $x : \bar{A}x = 0$ altérre).
Igazolnunk kell, hogy $Ax' = 0$ és $x' > 0$. Az $\bar{A}$ mátrixot A újraskálázásainak sorozatából kaptuk, tehát TA alakba írható, ahol

$$T = (I + v_k v_k^T) \dots (I + v_1 v_1^T)$$

olyan $v_1, \dots v_k \in \mathbb{R}^m$ vektorokra, melyek normája 1, tehát $Ax' = T^{-1}\bar{A}x' = 0$.
Igazolnunk kell, hogy $x' > 0$. Legyen $z \coloneqq A\bar{x}$, $z = T^{-1}\bar{A}\bar{x} = y$.
Tetszőleges $v \in \mathbb{R}^m$ 1-normájú vektorra $(I + vv^T)^{-1} = I - \frac{1}{2}vv^T$, így

$$T^{-1} = (I - \frac{1}{2}v_1v_1^T)\dots(I - \frac{1}{2}v_kv_k^T).$$

Figyeljük meg, hogy bármely $y \in \mathbb{R}^m$ vektorra és minden $v \in \mathbb{R}^m$ egységnormájú vektorra

$$\|(I - \frac{1}{2}vv^T)y\|^2 = \|y\|^2 - yvv^Ty + \frac{1}{4}vv^Tyvv^Ty = \|y\|^2 - \frac{3}{4}(v^Ty)^2 \leq \|y\|^2.$$

Ebből következik, hogy $\|z\| \leq \|\bar{y}\| < \delta$. Tehát

$$x' = \bar{x} - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{y} = \bar{x} - A^T(AA^T)^{-1}T^{-1}\bar{y} = \bar{x} - A^T(AA^T)^{-1}z > 0,$$

ahol az utolsó egyenlőtlenség abból adódik, hogy $\bar{x}_j \geq 1$, $j \in [n]$ és

$$|a_j^T(AA^T)^{-1}z| \leq \|(AA^T)^{-1}a_j\|\|z\| < \frac{1}{\delta}\delta = 1 \ \forall j \in [n].$$

2.5. Futási idő becslés

Minden koordinátaváltoztatás során $\|y\|$ az $(1 - \varepsilon^2)$ szorzótényezővel csökken. Amikor újraskálázunk, $\|y\|$ a 4-szeresére nő. Kezdetben $y = Ae$, ekkor $\|y\|^2 \leq 2^{4L}$. Ezekből következik, hogy k koordinátaváltoztatás után teljesül a $\delta^2 \leq \|Ae\|^2 4^N (1 - \varepsilon^2)^k \leq 2^{4L+2N} e^{-k\varepsilon^2}$ egyenlőtlenség. A 2.3.3. Állítás miatt $\delta \geq 2^{-2L}$, így

$$k \leq \varepsilon^{-2}(8L + 2N) = 121m^2(8 + 12m)L.$$

Így az iterációk száma az algoritmus futása során legfeljebb $N + k = O(m^3L)$.
Minden koordinátaváltoztató lépés során A^Ty kiszámolása $O(n)$ lépést vesz igénybe, feltéve, hogy az A^TA mátrixot már kiszámoltuk minden újraskálázás után. Az újraskálázó lépések száma $O(mL)$, $(I + \hat{y}\hat{y}^T)A$ kiszámítása $O(nm)$ elemi számolási lépést igényel, míg $A^T(I + \hat{y}\hat{y}^T)(I + \hat{y}\hat{y}^T)A$ kiszámolása $O(n^2)$-et, feltéve, hogy A^TA-t már kiszámoltuk. Így az algoritmus teljes műveletigénye $O((m^3n + mn^2)L)$.

2.6. Alkalmazás maximális tartójú megoldás keresése

Egy másik feladat, az (1) rendszer maximális pozitív koordinátájú megoldásának megkeresésére is használható az algoritmus. Jelölje S az $\mathcal{L}_+$ halmaz tartóját, $S := supp(\mathcal{L}_+)$. Az ismertetett algoritmus csak az $S = [n]$ esetet kezeli.
Egy $H \subseteq [n]$ indexhalmazra A_H jelölje A-nak azt a részmátrixát, mely a H-nak megfelelő oszlopokat tartalmazza. Jelölje $\alpha_j = \|a_j\|$ az eredeti mátrix i. oszlopvektorának hosszát. Az maximális tartóhalmaz keresésének algoritmusa a fejezetben ismertetett LP-algoritmust hívja meg többször az A mátrix oszlopainak részhalmazain az alábbi paraméterekkel:

$$\varepsilon := \frac{1}{12m^{3/2}} \ , \ N := 12m^2L \ , \ K = 2^{3mL}e^{N/4m} \tag{2.6}$$

Kezdetben az egész A mátrixon futtatjuk az alap algoritmust, majd elhagyjuk A-nak olyan oszlopait, melyek nem lehetnek benne az S halmazban. Az aktuális $H \subseteq [n]$ oszlophalmaz igazolhatóan tartalmazza az egész S-et.

Algorithm 2 Algoritmus maximális tartójú megoldás keresésére

Input: $A \in \mathbb{Z}^{m\times n}$, m-rangú mátrix, $\|a_j\| = \alpha_j$.

Output: Egy maximális tartójú megoldása az $Ax = 0$, $x \geq 0$ rendszernek.

Legyen $H = [n]$
while $H \neq \emptyset$ **do**
Futtassuk az 1-es algoritmust az új paraméterekkel
if egy x megoldást kapunk **then** STOP
Az output $S = H$, és az x megoldást kibővítjük $x_i = 0$-val minden $i \in [n] \setminus H$
end if
if egy $y \neq 0$, $A_H^T y \leq 0$ vektort ad az algoritmus **then**
Legyen $H = H \setminus i : a_i^T y > 0$ és futtassuk az algoritmust újra A_H-n
end if
if $\|y\| > \delta$ az 1-es algoritmus befejezésekor **then**
Legyen $H = i \in H : \|a_i\| < K\alpha_i$ és futtassuk az algoritmust újra A_H-n
end if
end while Output $S = \emptyset$.

Ha az LP-algoritmus a H halmazhoz talál egy teljes tartójú megoldást, akkor $S = H$, tehát a maximális tartójú megoldás az output. Ha egy nem nulla vektort talál, mely teljesíti az $A^T y \geq 0$ feltételt, akkor elhagyjuk H-ból az összes olyan i indexet, melyre $a_i^T y > 0$,

mivel ezek nem lehetnek S-ben. Ha az alap algoritmus egyik outputtal sem fejeződik be N újraskálázást követően, akkor megvizsgáljuk az újraskálázások utáni vektorok hosszát. Minden olyan i indexet kiveszünk S-ből, melynek hossza legalább K-szorosára nőtt az eredeti input méretéhez képest. Az algoritmus helyessége az alábbi tételen alapszik.

2.6.1. Tétel. *(bizonyítás nélkül) Tegyük fel hogy az 1-es algoritmus nem fejeződik be N újraskálázó lépés után, a* (2.6) *paraméterek használatával. Ekkor $\|a_i\| < K\alpha_i$ azt jelenti, hogy $i \notin S$.*

A tétel magába foglalja, hogy $S \subseteq H$ fennáll az algoritmus minden lépésekor. H dimenziója minden iteráció során legalább eggyel csökken, így az 1-es algoritmus m iteráción belül megáll.
Az alap algoritmus a módosított paraméterekkel $O(m^5L)$ futási idejő, így a maximális tartóhalmaz keresésének műveletigénye $O(m^6nL + m^2n^2L)$.
A 2.6.1 tétel háttere a következő. Ha a P_A politópot egy X altér tartalmazza, akkor térfogata 0. Ha az újraskálázó v vektor ebbe az altérbe esik, vagy merőleges vetülete nagy X-re, akkor arra következtethetünk, hogy P_A relatív térfogata jelentős mértékben nő. Ha v szinte merőleges X-re, akkor P_A csökkenhet, de csak kis mértékben. Ha egy a_i vektor hossza jelentősen megnő N újraskálázás után, akkor ez azzal ekvivalens, hogy az újraskálázó vektorok vetülete a_i irányába átlagosan nagy volt. $i \in S$ esetén ez azt jelenti, hogy az újraskálázás vektorának vetülete X-re átlagosan nagy volt és így P_A relatív térfogata meg kellett, hogy haladja az egységgömböt, ami ellentmondáshoz vezet. Ebből arra következtethetünk, hogy $i \notin S$.

3. fejezet

Dadush, Végh és Zambelli algoritmusa projekciós mátrixra

A fejezetben tárgyalt algoritmus [6] szintén a Dunagan-Vempala-féle koordinátaváltoztatást használja, viszont nem az A mátrixon, hanem az A mátrix $\mathcal{L}^\perp$-re való merőleges vetítésének mátrixán, Π-n.

3.1. A merőleges vetítés mátrixa

$\mathcal{L}^\perp$ az A mátrix sorai által generált altér $\mathbb{R}^n$-ben.

$$\mathcal{L}^\perp = \{z \in \mathbb{R}^n : \exists\, y \in \mathbb{R}^m \colon z = A^T y\}$$

A sorai $\mathcal{L}^\perp$ egy bázisát alkotják (mivel A rangja m), így $\mathcal{L}^\perp$ minden z eleme $y_1 a_1 + \dots y_m a_m$ alakba írható, $z = A^T y$ egy $y \in \mathbb{R}^m$ vektorra.
Egy x vektor $\mathcal{L}^\perp$-re való merőleges vetülete, $Proj_{\mathcal{L}^\perp}$ is $A^T y$ alakba írható, ezt az y vektort keressük. Bármely $x \in \mathbb{R}^n$ vektor egyértelműen felírható $x = Proj_{\mathcal{L}^\perp} x + Proj_{\mathcal{L}} x$ alakban, így

$$(x - Proj_{\mathcal{L}^\perp} x) \in \mathcal{L} \Rightarrow A(x - \underbrace{Proj_{\mathcal{L}^\perp} x}_{A^T y}) = 0$$

$$\begin{aligned} Ax - AA^T y &= 0 \\ Ax &= AA^T y \\ (AA^T)^{-1} Ax &= y \\ \Rightarrow Proj_{\mathcal{L}^\perp} x &= A^T (AA^T)^{-1} Ax \end{aligned}$$

Vezessük be a $\Pi = A^T(AA^T)^{-1}A$ jelölést. A Π mátrix i. oszlopát jelölje π_i, az i. sor j. elemét pedig π_{ij}.

A π projekciós mátrix alábbi tulajdonságait az elkövetkezendő tételek bizonyításánál ki fogjuk használni.

(i) $\forall x, z \in \mathbb{R}^n$ vektorra $\Pi x = 0 \Leftrightarrow x \in \mathcal{L}$ és $\Pi z = z \Leftrightarrow z \in \mathcal{L}^\perp$

(ii) $\Pi\Pi = (A^T(AA^T)^{-1})A)(A^T(AA^T)^{-1}A) = A^T I(AA^T)^{-1}A = \Pi$

(iii) $\forall w \in \mathbb{R}^n \|\Pi w\| \leq \|w\|$

(iv) (iii) következménye: $\forall j \in [m]$ indexre $\Pi j = \Pi e_j$, így $\|\Pi_j\| \leq 1$

(v) $\Pi_{jj} = \|\Pi_j\|^2 \; \forall j \in [n]$

(vi) $trace(\Pi) = \sum_{j=1}^{n} \|\Pi_j\|^2 = m$

3.2. Az LP algoritmus

Az algoritmusban az $\frac{1}{16n\sqrt{3m}}$ konstanst jelöljük ε-nal. Az $I \subseteq [n]$ indexhalmazhoz tartozó D_I diagonális mátrix legyen a következő: $d_{jj} = 1/2$, ha $j \in I$, különben $d_{jj} = 1$. Tehát $D_I = I - \frac{1}{2}\sum_{j \in I} e_j e_j$.

A vektorkoordináta-csökkentő lépés megegyezik a Dunagan Vempala által használttal, ám most az A mátrix helyett a Π mátrixon alkalmazzuk.

$$z' \coloneqq \mathbf{1}z - (\frac{\pi_k^T}{\|\pi_k\|}z) = z - \frac{\pi^T z \pi_k^T}{\|\pi_k\|^2} = z - \frac{z_k \pi_k}{\|\pi_k\|^2},$$

ahol az utolsó egyenlőségnél kihasználtuk, hogy $z \in \mathcal{L}^\perp$, így $z_j = \pi_j^T z \; \forall \; j \in [n]$. Ezáltal minden vektorváltoztató lépésnél az aktuális z normája $\sqrt{1-\varepsilon^2}$ szorzótényezővel csökken.

3.3. Az algoritmus végességének igazolása

A következő két lemma segítségével az algorimus helyességét igazoljuk: ha az output nem egy $\mathcal{L}_>$-beli pont, hanem egy az $\mathcal{L}_+$ tartójától különálló halmaz, akkor igazolandó, hogy a benne lévő indexek tényleg nem lehetnek a tartóban.
Egy $a \in \mathbb{R}$ számra legyen $a^+ \coloneqq max(0, a)$ és $a^- \coloneqq (-a)^+$.

Algorithm 3 Algoritmus 3

Input: Egy $A \in \mathbb{Z}^{m \times n}$ m rangú mátrix

Output: Egy $x \in \mathcal{L}_>$ megoldás, vagy egy $R \subseteq [n]$ indexhalmaz, mely nincs benne $\mathcal{L}_+$ tartójában.

Számítsuk ki a $\Pi = A^T(AA^T)^{-1}A$ mátrixot.

Legyen $w_j \coloneqq 1 \ \forall \ i \in [n], \ z \coloneqq \Pi w, \ count_j \coloneqq 0 \ \forall \ j \in [n]$.

while $count_j < L \ \forall \ j \in [n]$ **do**

if $w - z > 0$ **then**

STOP, output: $x \coloneqq w - z$

end if;

if $z \geq 0$ **then**,

STOP, output: $R \coloneqq j \in [n] : z_j \neq 0$

else legyen $i \coloneqq arg\min_{j \in [n]} \frac{z_j}{\|z\|\|\pi_j\|}$,

if $\frac{z_i}{\|z\|\|\pi_i\|} < -\varepsilon$ **then**, vektorváltoztatás

$z \coloneqq z - \frac{z_i \pi_i}{\|\pi_i\|^2}$; $w \coloneqq w - \frac{z_i e_i}{\|\pi_i\|^2}$

else újraskálázás

legyen $I \coloneqq \{j \in [n] : \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}}\}$;

$A \coloneqq AD_I$;

Számítsuk ki újra Π-t : $\Pi = A^T(AA^T)^{-1}A$;

Legyen $w_j \coloneqq 1 \ \forall j \in [n]$, $z \coloneqq \Pi w$;

$count_j \coloneqq count_j + 1 \ \forall \ j \in I$;

end if;

end if;

end while;

Output: $R \coloneqq \{j \coloneqq count_j = L\}$.

3.3.1. Lemma. *Legyen $z \in \mathcal{L}^\perp$ és legyen $k \in [n]$ olyan indenx, melyre $z_k > 0$. Ekkor minden $x \in \mathcal{L} \cap [0,1]^n$ pontra*

$$x_k \leq \frac{\sum_{j=1}^n z_j^-}{z_k}.$$

Bizonyítás. Tetszőleges $x \in \mathcal{L}$ pontra $z^T x = 0$, így $x_k = \frac{\sum_{j\in[n]\setminus\{k\}} -z_j x_j}{z_k}$. A lemma következik abból, hogy minden $x \in [0,1]^n$ vektorra $\sum_{j\in[n]\setminus\{k\}} -z_j x_j \leq \sum_{j=1}^n z_j^-$. □

3.3.2. Lemma. *A az algoritmus aktuális iterációjának alapmátrixa. Tegyük fel, hogy a jelenlegi $z = \Pi w$ kielégíti a $z_j \geq -\varepsilon\|z\|\|\pi_j\|$ feltételt, vagyis az algoritmusnak újraskálázást kell végrehajtania. Ekkor az $I = \{j \in [n] : \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}}\}$ halmaz nem üres. Továbbá, minden $x \in \mathcal{L} \cap [0,1]^n$ kielégíti az $x : k \leq \frac{1}{2}$ egyenlőtlenséget $\forall k \in I$ indexre.*

Bizonyítás.

$$\sum_{j=1}^n \Big(\frac{z_j^+}{\|z\|}\Big)^2 = \sum_{j=1}^n \Big(\frac{z_j}{\|z\|}\Big)^2 - \sum_{j=1}^n \Big(\frac{z_j^-}{\|z\|}\Big)^2 \geq 1 - \varepsilon^2 \sum_{j=1}^n \|\pi_j\|^2$$

Az első egyenlőtlenség abból következik, hogy z_j-t úgy választottuk, hogy $j = arg\min \frac{\pi_j^T z}{\|z\|\|\pi_j\|}$ minden j-re teljesül. Emiatt $\sum_{j=1}^n \left(\frac{z_j^-}{\|z\|}\right)^2 \leq \sum_{j=1}^n \varepsilon^2\|\pi_j\|^2$.
A második egyenlőségnél a Π mátrix nyomát, m-et helyettesítjük be kihasználva, hogy Π i. főátlóbeli eleme $\|\pi_j\|^2$.
A becslésből azt kapjuk, hogy

$$\sum_{j=1}^n \Big(\frac{z_j}{\|z\|}\Big)^2 \geq \sum_{j=1}^n \Big(\frac{z_j^+}{\|z\|}\Big)^2 > \frac{1}{3},$$

vagyis létezik legalább egy olyan $k \in [n]$, melyre $\frac{z_k}{\|z\|} > \frac{1}{\sqrt{3n}}$. A 3.3.1 felhasználásával arra jutunk, hogy az adott k indexre $\forall x \in \mathcal{L}\cap[0,1]^n$ vektorra teljesül a következő egyenlőtlenség:

$$x_k \leq \frac{\sum_{j=1^n z_j^-}}{z_k} \leq \varepsilon\frac{\|z\|}{z_k}\sum_{j=1}^n \|\pi_j\| \leq \varepsilon\sqrt{3n}\sqrt{n}\Big(\sum_{j=1}^n \|\pi_j\|^2\Big)^{\frac{1}{2}} = \varepsilon n\sqrt{3}\sqrt{m} = \frac{1}{16} < \frac{1}{2}$$

□

Vegyük észre, hogy az újraskálázás az A nullterét, $\mathcal{L}$-t a $D_I^{-1}\mathcal{L}$-re cseréli, vagyis minden $\mathcal{L}$-beli vektor I-ben lévő indexének megfelelő koordinátáit 2-vel szorozza. Legyen az inputmátrix nulltere $\mathcal{L}^0$. Az imént bizonyított lemmák megmutatják, hogy minden iteráció során $\mathcal{L}^0 \cap [0,1] \subseteq \{x \in \mathbb{R}^n : x_j < 2^{-\text{count}_j}\}$. Ha egy $j \in [n]$ az $\{Ax = 0,\, x \geq 0\}$ tartójában van, akkor létezik olyan x megoldása is az egyenlőtlenségrendszernek, melyre $x_j \geq 2^{-L}$ (5.6.1 , [13]). Tehát ha egy $j \in [n]$ indexre count_j meghaladja L-et, akkor j nem lehet benne a tartóhalmazban.

3.4. Futási idő becslés

Az algoritmus kezdetekor és minden újraskálázást követően $z = \Pi\mathbf{1}$, így $\|z\| \leq \|\mathbf{1}\| = \sqrt{n}$. Minden Dunagan-Vempala-féle átskálázó lépés a $\|z\|^2$-et $(1-\varepsilon)$ -szorosával csökkenti, és az algoritmus akkor fejeződik be, amikor $x := w - z > 0 \Leftrightarrow \|z\| < 1$. Ha két átskálázás között k számú koordinátacsökkentést hajtunk végre, akkor

$$n(1-\varepsilon^2)^k \geq 1$$

teljesül, átrendezve kapjuk, hogy $k \leq ln(n)\varepsilon^{-2} = O(n^2 m \log(n))$. Az algoritmus változónként legfeljebb L újraskálázást hajt végre, így koordinátacsökkentő lépésből legfeljebb $O(n^3 m \log(n) L)$ van. Minden ilyen lépés $O(n)$ műveletet igényel, így az algoritmus futási ideje $O(n^4 m \log(n) L)$.

3.5. Maximális tartóhalmaz keresése

A fejezetben ismertetett algoritmus az (1) rendszer megengedett megoldásainak tartóhalmazának megállapítására is használható. Ha az output egy olyan R indexhalmaz, mely nincs a tartóban, akkor legyen $x_j := 0$ minden $j \in R$ indexre, töröljük az A mátrix R-beli oszlopait, majd a kapott mátrixszal ismételjük meg az algoritmust. Ha befejeződik az algoritmus és egy $x > 0$ megengedett megoldást szolgáltat, akkor ez az x definiálja az eredeti feladat maximális tartóhalmazát. Ehhez legrosszabb esetben n-szer kell futtatnunk az algoritmust, így a maximális tartóhalmaz megadásának algoritmusa $O(n^5 mL)$ lépésben fut. Viszont ha úgy használjuk fel az ismertetett lineáris programozási algoritmust, hogy a count_j változókat nem nullázzuk le a procedúra meghívásakor, a futási idő $O(n^4 mL)$ lépésre csökken.

3.6. A poliéder térfogatváltozásának becslése

Legyen $Q_\Pi := \text{conv}(\Pi) \cap \text{conv}(-\Pi)$. Jelöje $\hat{\text{vol}}$ az $\mathcal{L}^\perp$-ben a térfogatot, $\hat{\text{vol}}(Q_\Pi)$ térfogatát fogjuk vizsgálni.

3.6.1. Lemma. *Legyen $\varepsilon' = 1/(16\sqrt{3n}m)$. Legyen $z \in \mathcal{L}^\perp$ olyan, $z_j \geq -\varepsilon'\|z\|\|\pi_j\|$ minden $j \in [n]$ indexre. Legyen I az a halmaz, hogy $I = j \in [n] : \frac{z_j}{\|z\|} > \frac{1}{\sqrt{3n}}$, és $\Pi' = D_I A^T (A D_I^2 A^T)^{-1} A D_I$. Ekkor*

$$\hat{vol}(Q_{\Pi'}) \geq e^{1/8} \hat{vol}(Q_\Pi).$$

Mivel $m \leq n$, így $\varepsilon \leq \varepsilon'$. Tehát ha az algoritmus újraskálázást hajt végre, az aktuális $z = \Pi w$ pontra teljesül 3.6.1, így az újraskálázást követően $\hat{\text{vol}}(Q_\Pi)$ konstans tényezővel csökken. Tudjuk, hogy $Q_\Pi \subseteq B(0,1) \cap \mathcal{L}$, így $\hat{\text{vol}}(Q_\Pi) \leq V_0$, ahol V_0 az m-dimenziós egységgömb térfogata.
Az előző fejezetben tárgyalt algoritmusnál a vizsgált poliéder térfogatának találtunk egy alsó korlátját abban az esetben, ha a poliéder nem üres, és ezt az algoritmus végességének bizonyításakor kihasználtuk. Ez a gondolatmenet itt is megállja a helyét.

3.6.2. Állítás. *Legyen $A \in \mathbb{Z}^{m\times n}$, jelölje L az A kódolási méretét, és legyen $\Pi = A^T(AA^T)^{-1}A$ Ha $\mathcal{L}_> \neq \emptyset$, akkor $\hat{vol}(Q_\Pi) \geq 2^{-3mL}$.*

Ebből következik, hogy ha az $Ax = 0$, $x > 0$ rendszernek van megoldása, akkor az algoritmus legfeljebb $(24 \ln 2)mL \in O(mL)$ újraskálázást hajthat végre. Mivel $m \leq n$,így ez azt jelenti, hogy hamarabb bebizonyosodhat, hogy az $Ax = 0$, $x > 0$ rendszernek nincs megoldása, mint hogy egy olyan j indexet találnánk, mely nincs benne a tartóban.

A fejezetben tárgyalt algoritmust Chubanov munkája inspirálta, melyet a következő fejezetben részletezünk. A két algoritmust ugyan azt az újraskálázást használja, és bizonyos tekintetben egymás duálisának tekinthető a két módszer. Chubanov a Neumann-féle koordináta-változtatást végzi el az $\mathcal{L}^\perp$ projekciós mátrixán, míg a Dadush-Végh-Zambelli algoritmusban a Dunagan-Vempala féle lépést használjuk $\mathcal{L}$ merőleges vetítésének mátrixán. A két módszer összehasonlítására Chubanov algoritmusának ismertetése után kerül sor.

4. fejezet

Chubanov algoritmusa projekciós mátrixra

Sergei Chubanov [11] 2013-ban publikált egy olyan lineáris programozási algoritmust, mely a homogén (1) egyenletrendszer megoldásai által meghatározott poliéder egy belső pontját adja meg, vagy tanúsítvánnyal szolgál a megoldás nemlétezéséről.
Ez a polinomiális időben futó algoritmus egy alap eljárást használ, melynek inputja a homogén egyenlőtlenségrendszer A alapmátrixa. Outputja egy szigorúan pozitív koordinátákból álló megoldás, vagy egy olyan információ, amivel az A mátrixot transzformálhatjuk annak reményében, hogy az (1) megoldásait az n dimenziós egységkockában a csupa 1-esből álló vektorhoz közelítsük.
Tekintsük az (1) egyenlőtlenségrendszert, ahol A egy $m \times n$-es racionális mátrix, rangja m. A megoldások poliédere egy $\mathbb{R}^n$-beli kúpot definiál.
Jelöljük P_A-val azt a mátrixot, megy az A mátrixnak az $\{x \in \mathbb{R}^n : Ax = 0\}$ altérre vett merőleges vetülete. Mivel az A nullterének ortogonális kiegészítő alterére való tükrözés mátrixa $A^T(AA^T)^{-1}A$, és minden $x \in \mathbb{R}^n$ egyértelműen felírható egy $\mathcal{L}$-beli és egy $\mathcal{L}^\perp$-beli vektor összegeként, így

$$P_A = I - A^T(AA^T)^{-1}A.$$

Tekintsük az (1) egyenlőtlenségrendszerrel ekvivalens $P_Ax > 0$ rendszert. Valóban egyenértékűek, hiszen ha x^* a $P_Ax > 0$ egyenőtlenség egy megoldása, akkor P_Ax^* (1) -nek egy pozitív megoldása, mivel $AP_Ax^* = 0$ annak következtében, hogy A oszlopvektorait $\mathcal{L}$-re levetítve nullvektorokat kapunk.
Az alap algoritmus pontok sorozatát konstruálja meg a következőképpen: egy $x^{(0)}$ nem

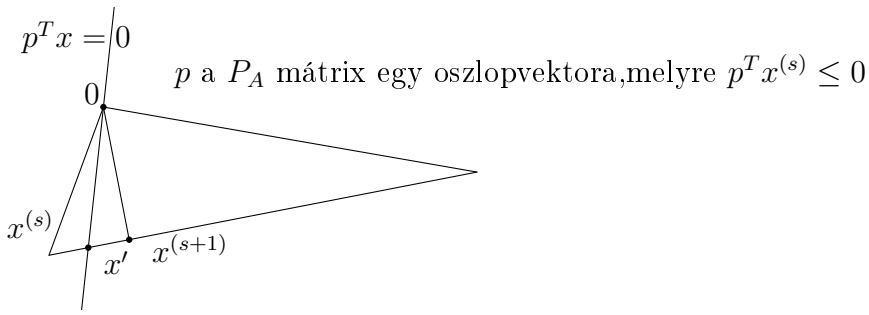
nulla pontból indulunk, mely a P_A oszlopainak egy konvex kombinációja ($x^{(0)} = P_A y^{(0)}$). Az s. iteráció során az $x^{(s)}$ pont kielégíti a $P_A x > 0$ egyenlőtlenséget, vagy megsérti azt, tehát $p^T x \leq 0$ a P_A mátrix i. sorára, p-re (mely megegyezik P_A i. oszlopával P_A szimmetriája miatt).
Legyen a pontsorozat $(s+1)$. tagja

$$x^{(s+1)} = \alpha x^{(s)} + (1-\alpha)p, \tag{4.1}$$

ahol az α számot úgy választjuk, hogy $x^{(s+1)}$ az origónak az $(x^{(s)}, p)$ szakaszra vett merőleges vetülete legyen.

$$\alpha = \frac{p^T(p - x^{(s)})^T}{\|p - x^{(s)}\|^2}$$

Ez a képlet adja meg az α konstanst, mely $p^T x^{(s)} \leq 0$ miatt a $[0,1]$ intervallumba esik.



4.1. ábra. Az alap eljárás egy iterációja

Az $x^{(s+1)}$ pont ugyanakkor a $(p, 0, x')$ pontok által meghatározott derékszögű háromszög magassága, ahol x' az $[x^{(s)}, p]$ szakasz és az $\{x : p^T x = 0\}$ hipersík metszéspontja (e metszéspont létezése a $p^T x^{(s)} \leq 0$ egyenlőtlenség teljesülése miatt igazolt.)
Az $(s+1)$. iteráció kezdőpontja tehát az alábbi alakban írható fel:

$$x^{(s)} = P_A (y^{(s)})^T : y^{(s)} \geq 0 : y^{(s)} \mathbf{1} = 1$$

Az algoritmus futása során végig fennmarad, hogy $x^{(i)}$ a P_A vektorok egy konvex kombinációja az őket összekötő képlet (4.1) miatt. Következésképpen az $(s+1)$. pont az $x^{(s+1)} = P_A (y^{(s+1)})^T$ képlettel számolható, ahol

$$y^{(s+1)} = \alpha y^{(s)} + (1-\alpha) e_i$$

.

4.0.1. Lemma. *Ha* $0 \notin [x^{(s)}, p]$, *akkor*

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x\|^2} + \frac{1}{\|p\|^2} \geq \frac{1}{\|x^{(s)}\|^2} + 1$$

Bizonyítás. Az egyenlőtlenséget elegendő csak abban a specális esetben bizonyítanunk, amikor $\|p - x'\| = 1$.
A Pitagorasz tételből adódik, hogy $1 = \|p - x'\|^2 = \|x'\|^2 + \|p\|^2$, hiszen az x' pont a p-re merőleges $\{p^T x = 0\}$ hipersíkban van.
Fejezzük ki $x^{(s+1)}$-et $x^{(s+1)} = x' + \lambda(p - x')$ alakban valamilyen λ-ra. Mivel $x^{(s+1)}$ a 0 merőleges vetülete az $[x^{(s)}, p]$ vonalra, és ezzel ekvivalensen az $[x', p]$ vonalra, így

$$0 = (x^{(s+1)})^T(p - x') = (x')^T(p - x') + [\lambda(p - x')]^T(p - x') = (x')^T p - \|x'\|^2 + \lambda,$$

kihasználva, hogy $p^T x' = 0$. Ekkor

$$\|x^{(s+1)}\|^2 = \|x'\| - 2\lambda\|x'\|^2 + \lambda^2\|p - x'\|^2 = \lambda - \lambda^2 = \|x'\|^2(1 - \|x'\|^2) = \|x'\|^2\|p\|^2$$

Tehát

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x'\|^2\|p\|^2} = \frac{\|p - x'\|^2}{\|x'\|^2\|p\|^2} = \frac{\|x'\|^2 + \|p\|^2}{\|x'\|^2\|p\|^2} = \frac{1}{\|x'\|^2} + \frac{1}{\|p\|^2}.$$

Vegyük észre, hogy $\|p\| \leq 1$, mivel P_A egy oszlopvektora p, illetve $\|x'\| \leq \|x^{(s)}\|$, hiszen x' a $0, x^{(s)}, p$ háromszög magassága. Feltéve, hogy a 0 nincs az $[x^{(s)}, p]$ egyenesen adódik, hogy

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x'\|^2} + \frac{1}{\|p\|^2} \geq \frac{1}{\|x^{(s)}\|^2} + 1.$$

Tehát az $\frac{1}{\|x^{(s)}\|^2}$ érték minden iteráció során legalább 1-gyel nő. □

4.1. Egy degenerált eset

Ha az origó vetülete az $[x^{(s)}, p]$ szakaszra önmaga, nem tudjuk új $x^{(s+1)}$ ponttal folytatni a sorozatot. Ebben az esetben $0 = \alpha x^{(s)} = (1 - \alpha)p$, vagyis $0 = \alpha P_A(y^{(s)})^T + (1 - \alpha)p$. Ekkor az $y = \alpha y^{(s)} + (1 - \alpha)e_i$ vektor megoldása az $yP_A = 0$ egyenletnek.
Tekinstük a homogén (1) rendszer egy x^* megoldását. Mivel $P_A x^* = x^*$, így $yP_A x^* = yx^* = 0$. Ekkor minden $k \in [n]$ indexre, amire $y_k > 0$, $x_k^* = 0$ teljesül. Az y vektor nem lehet az azonosan 0 vektor, hiszen y_i biztosan pozitív ($x^{(s)} \neq 0$, így $\alpha < 1$, tehát $(1 - \alpha) > 0$). Ekkor tehát nincs pozitív megoldása a homogén rendszernek. Továbbá

a $k \in [n]$ indexek egy olyan nem üres részhalmazához jutunk, melyre $x_k^* = 0$ minden x^* megoldására (1) -nek. A fenti eset a szigrúan pozitív x^* nemlétezésének egy másik igazolását is magába foglalja. Az y a P_A magterében van, így $y(I - A^T(AA^T)^{-1}A) = 0$, tehát $y = yA^T(AA^T)^{-1}A = zA$ alakba írható, ahol $z = yA^T(AA^T)^{-1}$. A Farkas lemma szerint az általunk megoldandó $Ax = 0$ egyenletnek nincs pozitív megoldása, ha az

$$y = zA$$
$$y \geq 0$$
$$y \neq 0$$

duál feladatnak létezik megoldása. A fenti y vektor éppen teljesíti mindhárom feltételt, így tanúsítványt kaptunk a minden koordinátájában pozitív x^* megoldás nemlétezésére.

4.2. Az alap eljárás

A lineáris programozási algoritmus egy alap eljárást hív meg többször, melynek célja az alábbi 3 eset valamelyikét reprezentáló pont vagy index megtalálása:

(i) Egy $k \in [n]$ index, melyre minden $x^* \in [0,1]^n$ megoldásra $x_k^* \leq \frac{1}{2}$;

(ii) Egy $k \in [n]$ index, melyre minden x^* megoldásra $x_k^* = 0$;

(iii) A lineáris megengedettségi feladat egy $x^* > 0$ megoldása.

Vegyük sorra azokat a megállási feltételeket, melyek a fenti eseteket garantálják.
Ha az aktuális x^* pont pozitív, akkor (iii) teljesül.
Ha $x_i^{(s)} \leq 0$ és $\mathbf{0}$ az $[x^{(s)}, p]$ szakaszon van, ahol p a P_A mátrix i. oszlopvektora, akkor a $k = i$ index teljesíti (ii)-t.
Maradt az (i) eset, melyhez két megállási feltétel is használható. Mivel P_A szimmetrikus, így $x^{(s)}$-et $x^{(s)} = (y^{(s)}P_A)^T$ alakban is megkaphatjuk. Legyen $x^* \in [0,1]^n$ az $Ax = 0$ egy megoldása, így $P_A x^* = x^*$ (ekkor $\|x^*\| \leq \sqrt{n}$). Legyen k az az index, melyre $y_k^{(s)} = \max y^{(s)}$. Mivel $y^{(s)}$ minden koordinátája nem negatív, így $y_k^{(s)}$ pozitív. A Cauchy-Schwartz egyenlőséget és x^* nemnegativitását kihasználva kapjuk, hogy

$$y_k^{(s)} x_k^* \leq y^{(s)} x^* = y^{(s)} P_A x^* ((x^{(s)})^T x^* \leq \|x^{(s)}\| \|x^*\| \leq \|x^{(s)}\| \sqrt{n}.$$

Ebből következik, hogy

$$x_k^* \leq \frac{\|x^{(s)}\| \sqrt{n}}{y_k^{(s)}}.$$

Vagyis az alábbi feltétel garantálja az (i) esetet:

$$y_k^{(s)} \geq 2\|x^{(s)}\|\sqrt{n}.$$

A lineáris megengedettségi feladat duálisának vizsgálatával egy másik (i)-re következtető feltételhez jutunk. Legyen ismét $y_k^{(s)} = \max y^{(s)}$ és tekintsük az alábbi optimalizálási feladatot:

$$\begin{aligned} \max x_k \\ Ax = 0, \\ 0 \geq x \geq 1. \end{aligned}$$

Ennek duális feladata így írható fel:

$$\begin{aligned} \min w\mathbf{1} \\ zA + w \geq e_k, \\ w \geq 0. \end{aligned}$$

Tekintsük az alábbi sorvektort:

$$z^0 = \frac{y^{(s)}A^T(AA^T)^{-1}}{y_k^{(s)}},$$

melyet A val jobbról szorozva ezt kapjuk:

$$z^0 A = \frac{y^{(s)}(I - P_A)}{y_k^{(s)}}.$$

Jelölje $[.]^+$ a vektor negatív koordinátáinak 0-ra cserélését. Ekkor

$$w^0 = \frac{[y^{(s)}P_A]^+}{y_k^{(s)}} = \frac{[x^{(s)}]^T}{y_k^{(s)}}$$

választással a (z^0, w^0) vektor a duális feladat egy megoldása. A dualitás tétel miatt ha $w^0\mathbf{1} \leq \frac{1}{2}$, akkor $x_k^* \leq \frac{1}{2}$ minden x^* megoldására az általunk vizsgált lineáris megengedettségi problémának. A $w^0\mathbf{1} \leq \frac{1}{2}$ feltétel (melyből (i) következik) az alábbi alakban is felírhatjuk, melyet az alap eljárás megállási feltételként fog használni:

$$\max y^{(s)} \geq 2 \cdot [(x^{(s)})^T]^+\mathbf{1}.$$

Algorithm 4 Az alap eljárás

Input: Egy $A \in \mathbb{R}$ mátrix és egy $y^{in} \geq \mathbf{0}$ sorvektor, melyre $y^{in}\mathbf{1} = 1$.

Output: (i), (ii), vagy (iii), az (i) esetben egy y^{out} sorvektor.

$y^{(0)} \coloneqq y^{in}$;

Számoljuk ki P_A-t és $y^{(0)}P_A$-t.

$x^{(0)} \coloneqq P_A(y^{(0)})^T$;

$s \coloneqq 0$;

while $\max y^{(s)} < 2 \cdot [(x^{(s)})^T]^+\mathbf{1}$ **do**

if $x^{(s)} > 0$ **then** Output: $x^{(s)}$;

elseLegyen $i \in [n]$ egy olyan index, melyre $x_i^{(s)} \leq 0$;

Legyen p a P_A mátrix i. oszlopa;

if 0 az $[x^{(s)}, p]$ szakaszon van **then** Output: $k \coloneqq i$, (ii) teljesül;

Számoljuk ki α-t;

$y^{(s+1)} \coloneqq \alpha y^{(s)} + (1-\alpha)e_i$;

$x^{(s+1)} \coloneqq \alpha x^{(s)} + (1-\alpha)p$;

$s \coloneqq s + 1$;

end if

end if

end while

Legyen k az az index, melyre $y_k^{(s)} = \max y^{(s)}$;

if $s \geq 1$ **then** (i) teljesül, $y^{out} = y^{(s-1)}$

else (i) teljesül, $y^{out} = \mathbf{0}$.

end if

4.2.1. Lemma. *Az alap eljárás legfeljebb $O(n^3)$ iterációból áll. Egy iterációja $O(n)$ lépésben fut.*

Bizonyítás. Mivel $y^{(s)}$ koordinátái nemnegatívak és összegük 1, így $\max y^{(s)} \geq \frac{1}{n}$. Ebből következik, hogy az (i)-hez tartozó megállási feltétel hamarabb bekövetkezik, mint az $\|x^{(s)}\| \leq \frac{1}{2n\sqrt{n}}$ egyenlőtlenség teljesülése. Mivel $\frac{1}{\|x^{(s)}\|^2}$ legalább 1-gyel nő minden iteráció során, így az utóbbi egyenlőtlenség legfeljebb $O(n^3)$ iteráció után teljesül, vagyis az alap eljárás legfeljebb ugyan ennyi műveletet igényel.
Minden s iteráció során az $y^{(s+1)}P_A$ vektor a már meglévő $y^{(s)}P_A$ és p vektorok konvex kombinációjaként számolható ki, $O(n)$ elemi lépésben. Így az alap eljárás while-ciklusának egy iterációja $O(n^3)$ nagyságrendű időben fut. □

4.2.2. Lemma. $\frac{1}{\|y^{out}P_A\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq 8Nn^2$.

Az alábbi jelöléseket használjuk a bizonyítás során: $y = y^{out}$ és $z = yA^T(AA^T)^{-1}$. Mivel $AP_A = 0$, így

$$(zA'' - yD)P_{A''} = -yDP_{A''}.$$

Figyelembe véve, hogy egy vektort $P_{A''}$-vel szorozva a vektor normája nem növekedhet (előző fejezet projekciós mátrixokra vonatkozó tulajdonságok (iii) pontja miatt) azt kapjuk, hogy

$$\|yDP_{A''}\| \leq \|zA''-yD\| = \|(zA-y)D\| \leq \|zA-y\| = \|yA(AA^T)^{-1}A-y\| = \|y(-P_A)\| = \|yP_A\|.$$

Kihasználva, hogy az y nemnegatív vektor koordinátáinak összege 1 adódik, hogy

$$1 - \sum_{i:D_{ii}<1} y_i \leq yD\mathbf{1}.$$

Az előbbi két becslést felhasználva a következőhöz jutunk:

$$\left(1 - \sum_{i:D_{ii}<1} y_i\right) \cdot \frac{1}{\|yP_A\|} \leq \frac{yD\mathbf{1}}{\|yDP_{A''}\|} = \frac{1}{\|y''P_{A''}\|}.$$

Mivel y a while-ciklus feltételét kielégíti, így

$$y_i \leq \max y < 2 \cdot [yP_A]^+\mathbf{1} \leq 2 \cdot \|yP_A\|\sqrt{n}$$

minen $i \in [n]$ indexre. Ebből pedig már következik a bizonyítandó állítás:

$$\frac{1}{\|yP_A\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq \left(2 \cdot \sum_{i:D_{ii}<1} y_i - \Big(\sum_{i:D_{ii}<1} y_i\Big)^2\right) \frac{1}{\|yP_A\|^2} \leq \frac{4N\sqrt{n}}{\|yP_A\|} \leq 8Nn^2.$$

(Az utolsó egyenlőtlenségnél kihasználtuk, hogy $\frac{1}{\|yP_A\|} \geq 2n\sqrt{n}$, különben a while-ciklus feltétele nem teljesülne.)

4.3. Az LP algoritmus

Az alap eljárás többszöri futtatása adja a lineáris programozási algoritmust, melynek lépéseit az alábbiakban részletezzük.

4.3.1. Futási idő megállapítása

A t. iteráció kezdetekor $A^{(t)} = AM^{(t)}$. Ez azt jelenti, hogy ha x^* az $A^{(t)}x = 0$ pozitív megoldása, akkor $M^{(t)}x^*$ az eredeti $Ax = 0$ egyenlet megoldása. Az LP algoritmus egy iterációját gyorsnak nevezzük, ha az alap eljárás (i)-t adja $y^{out} = 0$ -val. Ha egy iteráció (i) esettel végződik, de $y^{out} \neq 0$, akkor lassúnak nevezzük. Az utolsó iteráció nem biztos, hogy a 2 kategória valamelyikébe sorolható. Legyen t az LP algoritmus egy iterációja, t' pedig egy t előtti lassú iterációja úgy, hogy az össze t' és t közötti iteráció (ha volt ilyen) gyors volt. Legyen D a t iteráció kezdeti D mátrixa. Ekkor

$$M^{(t)} = M^{(t')}D \text{ és } A^{(t)} = A^{(t')}D.$$

Tehát a D mátrix megmutatja, hogy az utolsó lassú iteráció óta mely oszlopvektorok feleződtek.
Legyen T a gyors és lassú iterációk számának összege, F a gyors iterációk száma. Tudjuk, hogy $T \leq L_{min} + 1$.
Tekintsünk egy t lassú iterációt. Legyen y^{in} az alap eljárás inputjában lévő y^{in} vektor. Legyen y^{out} az alap eljárás outputja. Vezessük be az alábbi jelöléseket:

$$r'_t = \frac{1}{\|y^{in}P_{A^t}\|} \text{ és } r''_t = \frac{1}{\|y^{out}P_{A^t}}\|.$$

Ezzel a jelöléssel az alap eljárás while-ciklusainak száma legfeljebb

$$(r''_t)^2 - (r'_t)^2 + 1.$$

Az alap-algoritmus összes iterációszámának felső korlátja az LP algoritmus futása során

$$F + \sum_{\text{t lassú}} ((r''_t)^2 - (r'_t)^2 + 1) + 1.$$

Számozzuk a lassú iterációkat a π függvénnyel az LP algoritmusban szereplési sorrendjük alapján, vagyis reprezentálja $\pi(1), \ldots \pi(T - F)$ a lassú iterációkat. Az iterációk számát szeretnénk becsülni a bemeneti A mátrix méretével.

Algorithm 5 Az LP algoritmus

Input: Az $Ax = 0$, $x \geq 0$ lineáris megengedettségi probléma.

Output: A fenti rendszer egy $x > 0$ megoldása, vagy tanúsítvány arra, hogy ilyen megoldás nem létezik.

$t := 0$;

$y^{in} := \frac{\mathbf{1}}{n}$;

$y' = y^{in}$; (Az y' az előző iteráció y^{out} vektorát fogja tárolni $y^{out} \neq \mathbf{0}$ esetén.)

$M^{(t)} = I$; (Az $M^{(t)}$ mátrix az oszloptranszformációt hajtja vége.)

$D = I$; (A D mátrix az utolsó iteráció oszlopvektor változtatásait tárolja $y^{out} \neq \mathbf{0}$ esetén.)

$A^{(0)} := A$;

while $t \leq L_{min}$ **do** Hívjuk meg az alap eljárást $A^{(t)}$ és y^{in} outputtal;

if az eljárás (iii)-vel fejeződik be **then** Output: $x = M^{(t)}x^*$;

end if

if az eljárás (ii)-vel fejeződik be **then** return (ii);

end if

(Az alap eljárás (i)-vel végződik.)

Számoljuk ki $A^{(t+1)}$-et úgy, hogy $A^{(t)}$ k. oszlopát 2-vel osztjuk;

Számoljuk ki $M^{(t+1)}$-et úgy, hogy $M^{(t)}$ k. oszlopát 2-vel osztjuk;

if $y^{out} \neq \mathbf{0}$ **then**

$y' := y^{out}$;

$D = I$;

end if

Osszuk e 2-vel a D mátrix k. oszlopát;

$y^{in} := \frac{y'D}{y'D\mathbf{1}}$;

$t := t + 1$;

end while

$$\sum_{\text{t lassú}} ((r''_t)^2 - (r'_t)^2 + 1) = \sum_{q=1}^{T-F}((r''_{\pi(q)})^2 - (r'_{\pi(q)})^2 + 1) = \\ (r''_{\pi(T-F)})^2 - (r'_{\pi(1)})^2 + 1 + \sum_{q=1}^{T-F-1} ((r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 + 1) \tag{4.2}$$

Megjegyezzük, hogy $A^{(\pi(q+1))} = A^{(\pi(q))}D$, ahol D a $\pi(q+1)$ iteráció kezdeti D mátrixa. Jelöljük ennek a mátrixnak az 1-nél kisebb főátlómenti elemeinek számát N_q-val. Ekkoq

$$N_q \leq \pi(q+1) - \pi(q),$$

mivel az LP algoritmus minden iterációjával legfeljebb egy oszlopát változtatjuk meg az A^t mátrixnak. Tekintsük azt az y^{in} vektort, mely a $\pi(q+1)$ iteráció kezdetén az alap eljárás inputvektora. Ezt a vektort az $y^{in} = \frac{y^{out}D}{y^{out}D\mathbf{1}}$ képlettel számoltuk ki abból az y^{out} vektorból, melyet az LP algoritmus $\pi(q)$ iterációja során az alap eljárás adott. Most a D mátrix a $\pi(q+1)$ iteráció kezdeti D mátrixa, ahogy már fentebb írtuk, $A^{(\pi(q+1))} = A^{(\pi(q))}D$. A 2.3-as lemmát alkalmazzuk úgy, hogy A-t $A^{(\pi(q))}$-val, A''-t $A^{(\pi(q+1))}$ -el, y''-t y^{in}-nel, N-t pedig N_q-val helyettesítjük. Ekkor a legutolsó lemmából adódik, hogy

$$r''_{\pi(q)})^2 - r'_{\pi(q+1)})^2 \leq 8N_q n^2.$$

Figyelembe véve, hogy a $\sum_{q=1}^{T-F-1} N_q$ összeg legfeljebb T, a 2.3-as Lemmából azt kapjuk, hogy

$$\sum_{q=1}^{T-F-1} ((r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 + 1) \leq \sum_{q=1}^{T-F-1} (8N_q^2 T + 1) \leq 8n^2T + T.$$

Ekkor (4.2) -ből következik, hogy

$$\sum_{\text{t lassú}} ((r''_t)^2 - (r'_t)^2 + 1) \leq 4n^3 + 8n^2T + T + 1,$$

mivel $r''_{\pi(T-F)} \leq 2n\sqrt{n}$. Már csak az utolsó iteráció műveletigényét kell megemlítenünk, az alap eljárás legfeljebb $O(n^3)$ iterációt hajt végre ebben az esetben. Mindent összevetve azt kapjuk, hogy az alap-algoritmus összes iterációjának száma az LP algoritmus futása során $O(n^3+n^2L_{min})$ nagyságrendű. Az alap eljárás minden iterációja $O(n)$ lépésben fut a 2.2 lemma szerint. Az $x^{(0)}$ belső pont kiszámolása $O(n^2)$ nagyságrendű műveletet igényel. $P_{A^t)}$ kiszámítása legfeljebb $O(n^3)$ időben hajtható végre. Így a következő tételhez jutunk:

4.3.1. Tétel. *Az LP algoritmus futási ideje $O(n^4 + nL_{min})$.*

4.4. Kapcsolat Dadush, Végh és Zambelli algoritmusával

Az előző fejezetben ismertetett algoritmus az $\mathcal{L}^{\perp}$, míg Chubanov az $\mathcal{L}$ merőleges vetítésének mátrixával dolgozik. A két algoritmus újraskálázó lépése szinte megegyezik, és mindkét algoritmus egy $\mathcal{L}_{>}$ -beli pontot ad, ha egy ilyen létezik. A két algoritmus bizonyos értelemben viszont ellentétesen működik. A Neumann-féle lépések Chubanov algoritmusában az y nem negatív vektor $\mathcal{L}$-re való merőleges vetületének normáját csökkentik, amíg a Dunagan-Vempala-lépések a Dadush-Végh-Zambelli-algoritmusban csökkentik annak a vektornak a normáját, mely z-nek az $\mathcal{L}^{\perp}$ -re való merőleges vetülete. Tehát Chubanov iterációja $\|y\|^{-2}$ növekedését eredményezi, és az újraskálázás akkor következik be, ha $\|y\|$ elég kicsi. A Dadush-Végh-Zambelli-algoritmus pedig akkor fejeződik be, amikor $\|z\|$ elég kicsi ($\|z\| \leq 1$), és újraskálázást hajt végre, amikor egy koordináta-változtató lépést nem követné a $\|z\|$ jelentős csökkenése.

5. fejezet

A buborék algoritmus

Ebben a fejezetben egy polinomiális futási idejű algoritmusról [5] lesz szó, mely Chubanov [4] egy korábbi munkáján alapul. Az ismertetett lineáris programozási algoritmus a buborék algoritmus szubrutinon alapszik, mely az előző fejezetben tárgyalt alap algoritmushoz hasonló. Ám míg Chubanov alap algoritmusa szinte pontosan Neumann gondolatmenetét követi, a buborék algoritmus a relaxációs módszerhez köthető. Ez az algoritmus is az (1) rendszer egy megoldását keresi.
Chubanov algoritmusa homogenizálja az (1) rendszert, míg ez az algoritmus közvetlenül (1)-gyel dolgozik. További különbség, hogy Chubanov algoritmusa csak egy koordinátát változtat egy iteráció során, a buborék algoritmus egyszerre többet is. Illetve ahelyett, hogy az eredeti rendszert minden iteráció során megváltoztatnánk, Végh és Zambelli algoritmusa nem alakítja az A mátrixot, helyette a távolságok összehasonlításához használt normát módosítja.

5.1. Az LP algoritmus

Jelöljük P-vel az (1) lineáris megengedettségi feladat megoldásainak halmazát. Az A input mártix teljes sorrangú és $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$.
Legyenek $d_1, \ldots, d_m$ az (A, b) mátrix legnagyobb euklideszi normájú oszlopvektorai, Δ pedig jelölje ezen vektorok euklideszi normáinak szorzatát. Ekkor $\Delta < 2^L$ (bizonyítás a függelékben). Az (A, b) mátrix minden B négyzetes részmátrixára igaz, hogy $|detB| \leq \Delta$ (5.4) miatt. Ebből következik, hogy az (1) minden x megengedett megoldásához létezik $q \in \mathbb{Z}$, $1 \leq q \leq \Delta$ úgy, hogy $x_j = p_j/q_j$ valamilyen p_j egész számra, $0 \leq p_j \leq \Delta$, $j = 1, \ldots, n$. Következésképpen, ha $x_j > 0$, akkor $x_j \geq \Delta^{-1}$.

Az algoritmus fenntart egy $u \in \mathbb{R}^n$, $u > 0$ vektort úgy, hogy (1) minden bázismegoldása az $\{x : 0 \leq x \leq u\}$ n halmazba essen. Kezdetben $u_i \coloneqq \Delta$, $i \in [n]$.
Minden iteráció során az algoritmus megáll egy P-beli ponttal, vagy meghatároz egy $u' \in \mathbb{R}^n$, $0 < u' \leq u$ vektort úgy, hogy (1) minden x bázismegoldására $x \leq u'$ és van egy olyan $p \in [n]$ index, melyre $u'_p \leq u_p/2$. Ha $u_j \leq \Delta^{-1}$ egy $j \in [n]$ indexre, akkor a változók számát csökkenthetjük azzal, hogy x_j -t 0-ra állítjuk és az A mátrix j. oszlopát elhagyjuk.
Az LP algoritmus befejeződik, ha egy iteráció közben egy megengedett megoldást találunk, vagy ha az $Ax = b$ egyenletrendszernek egyértelmű megoldása van, vagy ha a lineáris megengedettségi feladat megoldhatatlan. Ha az egyértelmű megoldás nem negatív, akkor ez egy P-beli pontot ad, egyébként nincs megoldás.
Mivel minden iteráció során van egy p index, melyre u_p-t legalább a felére csökkentjük, és minden olyan x_j -re amit nem állítottunk 0-ra igaz, hogy $\Delta^{-1} \leq u_j \leq \Delta$, így az algoritmus legfeljebb $n \log(\Delta^2) \in O(nL)$ iteráció után megáll.

5.1.1. Tétel. *A Buborék algoritmus $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $u \in \mathbb{R}^n$, $u > 0$ inputok esetén $O(n^4)$ számú elemi lépés után az egyiket szolgáltatja:*

(i) Egy megengedett megoldását (1)*-nek;*

(ii) Egy $(u, w) \in \mathbb{R}^m \times \mathbb{R}^n_+$, $w \neq 0$, melyre $(v^T A + w^T)x < v^T b + \frac{1}{2n} w^T u$ minden $x \in \{x \in \mathbb{R}^n : 0 \leq x \leq u\}$ vektorra.

A buborék algoritmusról a következő szakaszban lesz szó.

5.1.2. Állítás. *Legyen $u \in \mathbb{R}^n$, $u > 0$ olyan, hogy* (1) *minden x bázismegoldására $x \leq u$ és legyen $(u, w) \in \mathbb{R}^m \times \mathbb{R}^n_+$ olyan vektorok, hogy $(v^T A + w^T)x < v^T b + \frac{1}{2n} w^T u$ minden $x \in \{x \in \mathbb{R}^n : 0 \leq x \leq u\}$ vektorra. Legyen $u'_j \coloneqq \min\{u_j, \frac{\sum_{j=1}^n u_i w_i}{2n w_j}\}$ $\forall$ $j \in [n]$ indexre. Ekkor* (1) *minden x bázismegoldására $x \leq u'$. Továbbá, ha p az az index, melyre $p \coloneqq \operatorname{argmax}_{j=1,\ldots,n}\{u_j w_j\}$, akkor $u'_p \leq u_p/2$.*

Bizonyítás. Mivel $w \neq \mathbf{0}$, így a (v, w) vektort átméretezhetjük úgy, hogy $\sum_{i=1}^n u_i w_i = 2n$ teljesüljön. Ezáltal $u'_j = \min\{u_j, w_j^{-1}\}$ $\forall$ $j \in [n]$. Tudjuk, hogy (1) minden x bázismegoldása kielégíti a $(v^T A + w^T)x < v^T b + \frac{1}{2n} w^T u$ egyenlőtlenséget, így $\forall$ $j \in [n]$ indexre

$$0 > v^T(ax - b) + \sum_{i=1}^n w_i(x_i - \frac{u_i}{2n} = \sum_{i=1}^n w_i x_i - 1 \geq w_j x_j - 1.$$

Ebből átszorzással adódik, hogy $x_j < w_j^{-1}$, így $x_j < u'_j$. Végül, mivel p-t úgy választottuk meg, hogy $p \coloneqq \operatorname{argmax}_{j=1,\ldots,n}\{u_j w_j\}$, így $u_p w_p \geq 2$, vagyis $w_p^{-1} \leq u_p/2$.

□

A fenti tétel és állítás magába foglalja, hogy a lineáris programozási algoritmus $O(n^5L)$ lépésben fut. A buborék algoritmus meghívásának száma $O(\lceil n/\log n\rceil L)$ nagyságrendűre csökkenthető, így az LP algoritmus műveletigénye $O(\lceil n^5/\log n\rceil L)$.

5.2. Skalárszorzással kapcsolatos képletek

Az alábbiakban ismertetett képletek és egyenlőtlenségek a buborék algoritmus skalárszorzáson alapuló újraskálázásának bevezetésére szolgálnak.

Egy D szimmetrikus, pozitív definit mátrix esetén $\langle x,y\rangle_D = x^TDy$ jelöli x és y D-skalárszorzatát. Az $\|x\|_D = \sqrt{\langle x,x\rangle_D}$ képlettel definiált normát D-normának nevezzük, és $\|.\|_D$ -val jelöljük. Az x és y pontok D -távolsága $\|x-y\|_D$. Egy $c\in\mathbb{R}^n$ középpontú és $r>0$ sugarú D-gömböt a $B_D(c,r) \coloneqq \{x : \|x-c\|_D \le r\}$ képlet definiál.
Egy adott x pontra, az az $y \in \{x\in\mathbb{R}^n : Cx = d\}$ pont, mely x-hez a D-normában legközelebb van, az alábbi képlettel írható le:

$$y = x + D^{-1}C^T(CD^{-1}C^T)^{-1}(d-Cx), \tag{5.1}$$

így a D-távolság az x pont és az $\{x\in\mathbb{R}^n : Cx=d\}$ poliéder között

$$\|y-x\|_D = \sqrt{(d-Cx)^T(CD^{-1}C^T)^{-1}(d-Cx)}.$$

Lássuk, hogyan jöhet ki ez a képlet. Elegendő a homogén esetre belátnunk a képlet helyességét, így legyen $W=\{x : Cx=0\}$ a C nulltere, az $\{x\in\mathbb{R}^n : Cx=d\}$ halmaz egy eltoltja. Keressük meg egy x ponthoz D-legközelebbi pontot W-ben.
W egy bázisát jelölje $\{b_1,\dots b_k\} = B$, a W ortogonális kiegészítő alterét jelölje $W_\perp$, ennek bázisa legyen $b'_{k+1},\dots b'_n$. Az x vektor egyértelműen felírható $x = \beta_1b_1+\ldots+\beta_kb_k + \beta_{k+1}b'_{k+1}+\ldots\beta_nb'_n$ alakban.
Keressük meg azt az M mátrixot, mely az x vektor W-re merőleges alkotóját adja meg, vagyis $x_{W_\perp} = Mx$. Ha $x\in W$, akkor $Mx=0$, mivel x csak x_W-ből áll. Ha $x\perp W$, akkor $Mx=x$ (a vektor csak $x_{W_\perp}$ -ből áll). Tudjuk, hogy C^T minden oszlopvektora merőleges W-re, így az előbbi megállapítás miatt $MC^T = C^T$.
Az M mátrixnak olyannak kell lennie, hogy W bázisára, B-re merőleges, tehát $Mb_i = 0$ minden $i\in[k]$ indexre, vagyis $MB=0$. Tudjuk, hogy $CB=0$. Mivel $C^T(CC^T)^{-1}CC^T = C^T$, így $M = C^T(CC+T)^{-1}C$ teljesíti az $MC^T=C^T$ feltételt. (CC^T inverze létezik, mivel

C teljes sorrangú.) $C^T(CC^T)^{-1}CB = 0$ kihasználva, hogy $CB = 0$, így ez az M-választás $MB = 0$-t is teljesíti. Tehát $x_{W_\perp} = C^T(CC^T)^{-1}Cx$, $x_W = x - C^T(CC^T)^{-1}Cx$. Most keressünk egy olyan $\{f_1, \ldots, f_k\} = F^T$ mátrixot, hogy F a B bázist alkotó vektorokra D-merőleges, tehát

$$\langle f_i, b_j \rangle_D = 0 \; \forall \; i, j, \in [1, \ldots, k] \Leftrightarrow f_i D b_j = 0 \; \forall \; i, j, \in [1, \ldots, k] \Leftrightarrow F^T DB = 0$$

Tudjuk, hogy $C^T B = 0$, így legyen $F^T D = C^T$, tehát $F = D^{-1}C$ választás jó lesz.
M és F választása miatt $MF^T = F^T$, tehát $MD^{-1}C^T = D^{-1}C^T$, melyből átrendezéssel adódik a keresett képlet:

$$M = D^{-1}C^T(CD^{-1}C^T)^{-1}C.$$

5.2.1. Megjegyzés. Ha y az a pont, mely x-től minimális D-távolságra van az $\{x \in \mathbb{R}^n : Cx = d\}$ poliéderben, akkor a $\lambda \coloneqq (CD^{-1}C^T)^{-1}(d - Cx)$ egyértelmű megoldása az $y - x = D^{-1}C^T\lambda$ és $\|y - x\|_D^2 = (d - Cx)^T\lambda$ egyenleteknek.

Adott $\alpha \in \mathbb{R}^n \setminus \{0\}$ vektorra és $\beta \in \mathbb{R}$ számra az (5.1) képletet alkalmazva az az y, mely az $\{x \in \mathbb{R}^n : \alpha^T x = \beta\}$ poliédertől minimális D-távolságban van

$$y = x + \frac{D^{-1}\alpha(\beta - \alpha^T x)}{\alpha^T D^{-1}\alpha} \text{ és } \|y - x\|_D = \frac{|\beta - \alpha^T x|}{\sqrt{\alpha^T D^{-1}\alpha}}. \tag{5.2}$$

Az alábbi lemmában ismertetett egyenlőtlenségeket a bizonyítások során kihasználjuk.

5.2.2. Lemma. *Legyen $K \subseteq \mathbb{R}^n$ egy konvex halmaz, $x' \in \mathbb{R}^n \setminus K$ és z az a K-beli pont, mely x'-től minimális D-távolságra van. Ekkor*

$$\langle z - x', x - x' \rangle \geq \|z - x'\|_D^2$$

minden $x \in K$-ra, és

$$\langle z - x', x - x' \rangle \leq \|z - x'\|_D^2$$

minden $x \in B_D(x', \|z - x'\|_D)$ pontra.

5.3. A buborék algoritmus

Adott $u \in \mathbb{R}^n$, $u > 0$ vektor esetén legyen $l \coloneqq \frac{u}{2n}$ és $\tilde{P} \coloneqq \{x : Ax = b, x \geq l\}$. Jelölje $D = (d_{ij})$ azt az $n \times n$-es diagonális mátrixot, melynek i-edik főátlóbeli eleme $d_{ii} = 4u_i^{-2}$. Mivel $u > 0$, így D pozitív definit. Továbbá, $\{x \in \mathbb{R}^n : 0 \leq x \leq u\} \subset B_D(0, 2\sqrt{n})$ kihasználva, hogy $\sqrt{x^T dx} \leq 2\sqrt{n}$.
Legyen $\mathcal{A} \coloneqq \{x \in \mathbb{R}^n : Ax = b\}$.

Algorithm 6 Buborék algoritmus

Input: Az $Ax = b$, $x \geq 0$ egyenletrendszer és az $u > 0$ vektor.

Output: Az alábbiak közül az egyik:

1. Egy P-beli pont
2. $(v, w) \in \mathbb{R}^m \times \mathbb{R}^m_+$ vektorok, melyekre
   $(v^T A + w^T)x < v^T b + w^T l \ \forall x \in B_D(0, 2\sqrt{n})$.

Legyen z a 0-hoz D-legközelebbi pont $\mathcal{A}$-ban. Legyen $l \coloneqq \frac{u}{2n}$.
while $\|z\|_D \leq 2\sqrt{n}$ **do**
  if $z \in P$ **then** STOP;
  else Legyen i olyan index, melyre $z_i < 0$;
    Legyen $K \coloneqq \{x \in \mathcal{A} : \langle z, x\rangle \geq \|z\|_D^2,\ x_i \geq l_i\}$.
    if $K = \emptyset$ **then** output: $(v, w) \in \mathbb{R}^m \times \mathbb{R}^m_+$ úgy, hogy
      $A^T v + w = 0$ és $v^T b + v^T l > 0$;
    else Állítsuk át z-t arra a pontra, mely K-ban minimális D-távolságra van 0-tól
    end if
  end if
end while
Az output $(v, w) \in \mathbb{R}^m \times \mathbb{R}^m_+$ úgy, hogy
$Dz = A^T v + w$ és $\|z\|_D^2 = v^T b + w^T l$.

5.3.1. Lemma. *Legyen $z \in \mathcal{A}$ olyan, hogy $\langle z, x\rangle_D \geq \|z\|_D^2$ fennáll minden $x \in \tilde{P}$-beli pontra. Ha $z \notin P$, akkor legyen $i \in [n]$ olyan, hogy $z_i < 0$ és definiáljuk K-t a $K \coloneqq \{x \in \mathcal{A} : \langle z, x\rangle_D \geq \|z\|_D^2,\ x_i \geq l_i\}$ egyenlőtlenségekkel. Tegyük fel, hogy $K \neq \emptyset$ és legyen z' az origóhoz D-legközelebbi pont K-ban. Ekkor $\langle z', x\rangle_D \geq \|z'\|_D^2$ teljesül minden $x \in \tilde{P}$ pontra és $\|z'\|_D^2 > \|z\| + \frac{1}{n^2}$.*

Bizonyítás. Mivel K egy olyan poliéder, mely tartalmazza $\tilde{P}$-ot, így 2.6.1 -et alkalmazva $\tilde{P}$-ra kapjuk, hogy $\langle z', x\rangle_D \geq \|z'\|_D^2$ minden $x \in \tilde{P}$ pontra. Alkalmazva (5.2)-et $\alpha = e_i$, $\beta = l_i$, és $x' = z$ helyettesítéssel kapjuk, hogy a z és $\{x \in \mathbb{R}^n : x_i = l_i\}$ közötti D-távolság

$$\frac{|l_i - z_i|}{\sqrt{(e_i)^T d^{-1} e_i}} > 2\frac{l_i}{u_i} = \frac{1}{n}.$$

Ebből következik, hogy a $B_D(z, \frac{1}{n})$ gömb minden pontja megsérti az $x_i \geq l_i$ egyenlőtlenséget.
Most lássuk be, hogy minden olyan $x \in B_D\left(0, \sqrt{\|z\|_D^2 + \frac{1}{n^2}}\right)$ pont, melyre $\langle z, x\rangle_D \geq \|z\|_D^2$ igaz, hogy benne van a $B_D(z, \frac{1}{n})$ gömbben. Valóban, minden ilyen x pontra

$$\|z - x\|_D^2 = \|z\|_D^2 + \|x\|_D^2 - 2\langle z, x\rangle_D \leq 2\|z\|_D^2 + \frac{1}{n^2} - 2\|z\|_D^2 = \frac{1}{n^2}.$$

Mivel minden $x \in B_D(z, \frac{1}{n})$ pontra $x_i < l_i$, így $B_D\left(0, \sqrt{\|z\|_D^2 + \frac{1}{n^2}}\right)$ és K metszete üres. Így z' definíciója miatt $\|z'\|_D^2 > \|z\|_D^2 + \frac{1}{n^2}$. $\square$

Ha a $\langle z, x\rangle_D \geq \|z\|_D^2$ lineáris következménye az $\{Ax = b,\ x \geq l\}$ rendszernek, akkor léteznek olyan $(v, w) \in \mathbb{R}^m \times \mathbb{R}_+^m$ vektorok, hogy $Dz = A^T v + w$ és $\|z\|_D^2 \geq v^T b + w^T l$. Szorozzuk be a $Dz = A^T v + w$ egyenlet mindkét oldalát balról x-el, ekkor azt kapjuk, hogy $\langle z, x\rangle_D = v^T b + w^T x$ kihasználva, hogy $\langle z, x\rangle = (Dz)^T x$. Ekkor

$$v^T b + w^T x = \langle z, x\rangle_D \leq \|z\|_D^2 = v^T b + w^T l,$$

és mivel $x \geq l$ és $w \geq 0$, így az egyenlőtlenségnek egyenlőséggel kell teljesülnie. Így léteznek olyan $(v, w) \in \mathbb{R}^m \times \mathbb{R}_+^m$ vektorok, hogy $Dz = A^T v + w$ és $\|z\|_D^2 = v^T b + w^T l$. Ha a K halmaz az 5.3.1 lemmában üres, akkor a Farkas lemma miatt létezik egy $(v, w) \in \mathbb{R}^m \times \mathbb{R}_+^m$ vektor, melyre $A^T v + w = 0$ és $v^T b + w^T l > 0$. A buborék algoritmus outputja ez a (v, w) vektor, melynek kiszámítása a következő alfejezet témája.

Az 5.3.1 lemma miatt minden iteráció során $\|z\|_D^2$ legalább $\frac{1}{n^2}$-tel nő. Így legfeljebb $4n^3$ iterácó után $\|z\| > 2\sqrt{n}$. Következésképpen, ha az algoritmus a while-cikluson kívül áll meg, akkor a $\langle z, x\rangle_D \geq \|z\|_D^2$ egyenlőtlenség minden $\tilde{P}$-beli x-re teljesül, illetve minden

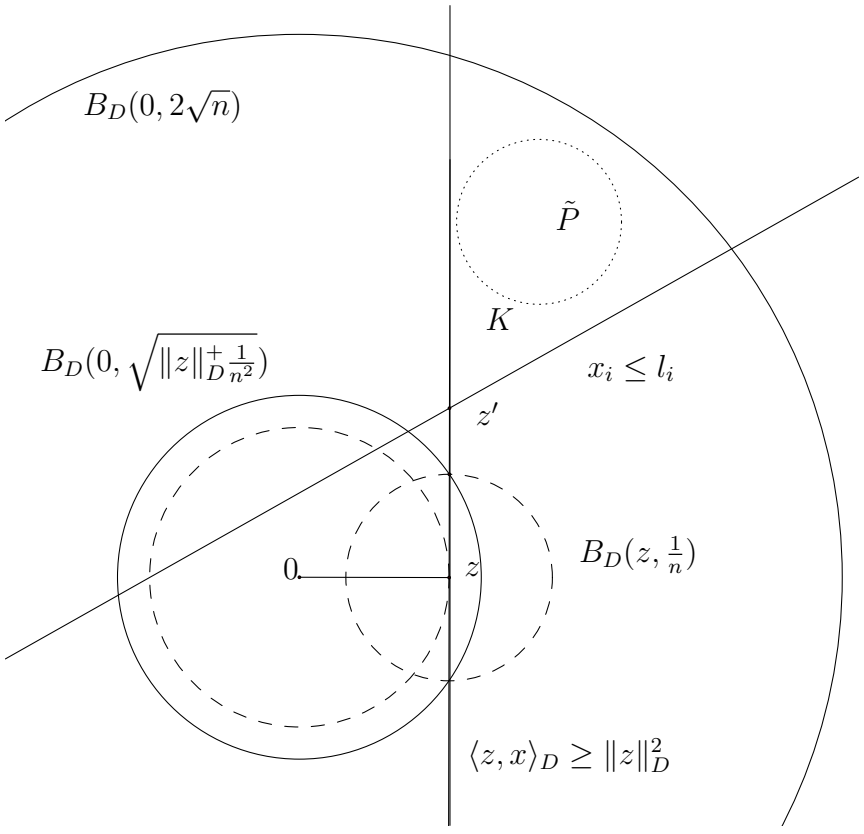
$B_D(0, 2\sqrt{n})$-beli pontra, így minden $\{x : 0 \leq x \leq u\}$-beli x-re nem teljesül. Ez az 5.1.1 második kimenetelének felel meg.

Ahhoz hogy megmutassuk, hogy a buborék algoritmus polinomiális, két dolgot kell vizsgálnunk. Az egyik, hogy hogyan számoljuk ki egy iteráció során az új z pontot és a

$$(v, w) \in \mathbb{R}^m \times \mathbb{R}^m_+$$

vektorokat úgy, hogy $Dz = A^T v + w$ és $\|z\|_D^2 = v^T b + w^T l$ teljesüljenek. A másik feladatunk megmutatni, hogy a $\tilde{P}$ poliéder üres, amikor a buborék algoritmus a $K = \emptyset$ feltétel teljesülése miatt áll meg.

A buborék algoritmus egy P-beli ponttal fejeződik be, vagy egy olyan (v, w) vektort ad, mellyel u-t újradefiniálhatjuk, hogy az eljárást ezzel az új input vektorral futtassuk. Ez utóbbi két esetben következhet be: ha $K = \emptyset$ egy iteráció közben, vagy egy while-ciklus végén.



5.1. ábra. A $B_D(0, \sqrt{\|z\|_D^2 + \frac{1}{n^2}})$ gömb és a K halmaz metszete üres

5.4. A következő z pont kiszámítása

Azokat a $(v, w) \in \mathbb{R}^m \times \mathbb{R}^m_+$ vektorokat keressük, melyekre $Dz = A^T v + w$ és $\|z\|_D^2 = v^T b + w^T l$. Az alábbiakban részletezzük ezen vektorok kiszámítását.
Jelölje r^0 azt az $\mathcal{A}$-beli pontot, mely az origóhoz D-legközelebb van. Legyen $v^0 \in \mathbb{R}^n$ az a vektor, melyre $Dr^0 = A^T v^0$ és $\|r^0\|_D^2 = b^T v^0$.

5.4.1. Megjegyzés. Vegyük észre, hogy $\forall x \in \mathcal{A}$, $\langle x - r^0, r^0 \rangle_D = 0$ (mivel $r^0 \in \mathcal{A}$), így $\|x\|_D^2 = \|x - r^0\|_D^2 + \|r^0\|_D^2$. Ebből következik, hogy minden $C \subseteq \mathcal{A}$ konvex halmazra C-ben az origóhoz D-legközelebbi pont megegyezik C-ben az r^0-hoz D-legközelebbi ponttal.

Feltételezhetjük, hogy $\emptyset \neq \{x \in \mathcal{A} : x_j \geq l_j\} \subset \mathcal{A}$. Emiatt létezik egy olyan $\alpha^j \in \{\mathcal{A} \setminus r^0\}$ vektor és egy $\beta_j \in \mathbb{R}$ szám, hogy $\|\alpha_j\|_D = 1$ és

$$\{x \in \mathcal{A} : x_j \geq l_j\} = \{x \in \mathcal{A} : \langle \alpha^j, x \rangle_D \geq \beta\}$$

(α^j, β_j) Gauss-eliminációval számolható a $D\alpha^j = A^T v^j + w^j e_j$ és $\beta_j = b^T v^j + w^j e_j$ egyenletekből. Jelölje r^j azt a pontot, mely az $\{x \in \mathcal{A} : x : j = l_j\}$ halmaznak az origóhoz D-legközelebbi pontja. Ekkor $\|\beta_j\| = \|r^j - r^0\|_D$ és $r^j = r^0 + \beta \alpha_j$.
Tegyük fel, hogy $r^0 \notin P$, különben az algoritmus rögtön megállna. Tehát az első iteráció során $z = r^t$ egy olyan t indexre, melyre $\beta_t > 0$. Az első iterációnál válasszuk t-t úgy, hogy $t = \text{argmax}_j \beta_j$. Így $\|z\|_D \geq \|r^j\|_D \ \forall j \in [n]$, melyre $\beta_j > 0$. 5.3.1 és 5.4.1 miatt $\|z - r^0\|_D \geq \frac{1}{n^2}$.
Minden iteráció során adott egy $z \in \mathcal{A} \setminus P$ pont, melyre $\langle z, x \rangle_D \geq \|z\|_D^2$ minden $x \in \tilde{P}$ pontra. Legyen $\alpha = (z - r^0)/\|z - r^0\|_D$ és $\beta = \|z - r^0\|$. Ekkor

$$\{x \in \mathcal{A} : \langle z, x \rangle_D \geq \|z\|_D^2\} = \{x \in \mathcal{A} : \langle \alpha, x \rangle \geq \beta\}.$$

Fenntartunk egy olyan $\lambda \in \mathbb{R}^n_+$ vektort, melyre

$$(\alpha, \beta) = \sum_{j=1}^{n} \lambda_j (\alpha^j, \beta_j).$$

Az így tárolt változók biztosítják azokat a (v, w) vektorokat, melyekre $Dz = A^T v + w$ és $\|z\|_D^2 = v^T b + w^T l$ az alábbi definíciókkal:

$$v \coloneqq v^0 + \beta \sum_{j=1}^{n} \lambda_j v^j \text{ és } w \coloneqq \beta \sum_{j=1}^{n} \lambda_j w_j e_j.$$

Minden iteráció során az algoritmus befejeződik, ha $z \geq 0$, vagy veszünk egy i indexet, melyre $z_i < 0$ és definiáljuk K-t : $K \coloneqq \{x \in \mathcal{A} : \langle z, x\rangle \geq \|z\|_D^2,\, x_i \geq l_i\}$.
Ha $K = \emptyset$, akkor az algoritmus befejeződik. Egyébként az aktuális z-t lecseréljük egy $z' \in K$ pontra, mely az origótól minimális D-távolságra van.

Ebben a részben leírjuk, hogyan számolható ki z', ha $K \neq \emptyset$, illetve hogy $K = \emptyset$ esetén miért bizonyosodhatunk meg a lineáris megengedettségi feladat megoldásának nemlétezéséről.
Legyen

$$\bar{K} \coloneqq \{x \in \mathbb{R}^n : \langle \alpha, x\rangle_D \beta,\, \langle \alpha^i, x\rangle_D \geq \beta_i\}.$$

Ezzel a jelöléssel $K = \mathcal{A} \cap \bar{K}$.

5.4.2. Állítás. *Ha $\bar{K} \neq \emptyset$, akkor a 0-hoz D-legközelebbi pont $\bar{K}$-ban megegyezik z'-vel. Következésképpen, $K \neq \emptyset$ akkor és csakis akkor, ha $\bar{K} \neq \emptyset$.*

Bizonyítás. Legyen $\bar{z}$ a $\bar{K}$-ban az r^0-hoz D-legközelebbi pont. Mivel $\langle \bar{z} - r^0, x\rangle_D \geq \|\bar{z} - r^0\|_D^2$ teljesül minden $x \in \bar{K}$ pontra, így léteznek olyan $\mu_1, \mu_2 \geq 0$ számok ((5.3)) , hogy $D(\bar{z} - r^0) = \mu_1 D\alpha^i + \mu_2 D\alpha$. Ebből kapjuk, hogy $\bar{z} = r^0 + \mu_1\alpha + \mu_2\alpha$, melyből következik, hogy $A\bar{z} = Ar^0 + \mu_1 A\alpha^i + \mu_2 A\alpha = b$. Ez igazolja, hogy $\bar{z} \in \mathcal{A}$, és így $\bar{z} \in K$. Mivel $K \subseteq \bar{K}$, így $\bar{z}$ K-ban az r^0-hoz D-legközelebbi pont, és így az origóhoz legközelebbi is, tehát $z' = \bar{z}$.
Következésképpen ha $\bar{K} \neq \emptyset$, akkor $K \neq \emptyset$. Visszafelé pedig ha $K \neq \emptyset$, akkor $\bar{K} \neq \emptyset$, mivel $K \subseteq \bar{K}$. $\square$

5.4.3. Állítás. *$K \neq \emptyset$ akkor és csakis akkor, ha α^i és α lineárisan függetlenek. Ha $K \neq \emptyset$, akkor z' az*

$$\mathfrak{L} \coloneqq \{x : \langle \alpha, x\rangle_D = \beta,\, \langle \alpha^i, x\rangle_D = \beta_i\}$$

halmaz azon pontja, mely r^0-hoz D-legközelebb van.

Bizonyítás. Tegyük fel, hogy $K \neq \emptyset$. 5.4.2 miatt z' a $\bar{K}$.ban az a pont, mely r^0-hoz D-legközelebb van. Meg kell mutatnunk, hogy z' az $\mathfrak{L}$-ben az r^0-hoz D-legközelebbi pont. Ehhez elegendő belátni, hogy z' kielégíti az $\langle \alpha, z'\rangle_D = \beta$ és $\langle \alpha^i, z'\rangle_D = \beta_i$ egyenleteket.
Indirekt tegyük fel, hogy $\langle \alpha^i, z'\rangle_D > \beta_i$, ekkor z' az $\{x : \langle \alpha, x\rangle_D \geq \beta\}$ halmaz r^0-hoz D-legközelebbi pontja. Ekkor $z' = z$, és ez ellentmondáshoz vezet, mivel $\|z'\|_D > \|z\|_D$.

Ha $\langle \alpha, z' \rangle_D = \beta$, akkor z' az $\{x : \langle \alpha^i, x \rangle_D \geq \beta_i\}$ halmaz r^0-hoz D-legközelebbi pontja. Ha $\beta_i > 0$, akkor $z' = r^i$ ami ellentmond annak, hogy $\|z'\|_D > \|z\|_d \geq \max_j \beta_j$. Így ha $\beta_i \leq 0$, akkor $z' = r^0$, ami nem lehet, mivel $r^0 \notin \bar{K}$.
Az állítás első részének igazolásához felhasználjuk 5.4.2 következményét, vagyis hogy $K \neq \emptyset$ akkor és csakis akkor, ha $\bar{K} \neq \emptyset$. $\bar{K}$ definíciója miatt $\bar{K} \neq \emptyset$ akkor és csakis akkor, ha α és α lineárisan függetlenek. Tegyük fel, hogy $\bar{K} \neq \emptyset$. Ha α^i és α lineárisan függenének, akkor $\mathfrak{L} \coloneqq \{x : \langle \alpha^i, x \rangle_D = \beta_i\} = \{x : \langle \alpha, x \rangle_D = \beta\}$. Mivel z' az $\mathfrak{L}$-ben az r^0-hoz D-legközelebbi pont, így $z' = z$, ami ellentmondás. $\square$

Vizsgáljuk meg azt az esetet, amikor $K \neq \emptyset$.
5.4.3 miatt z' az $\mathfrak{L}$ halmaz r^0-hoz legközelebbi pontja, illetve α^i és α lineárisan függetlenek. 5.2.1 szerint $z' - r^0 = \mu_1 \alpha + \mu_2 \alpha^i$, ahol $(\mu_1, \mu_2)^T = (CD^{-1}C^T)^{-1}(d - Cr^0)$, ahol C az a $2 \times n$-es mátrix, melynek sorai $(D\alpha^i)^T$ és $D\alpha^T$, illetve $d \in \mathbb{R}^2$ egy olyan vektor, melyben $d_1 = \beta_i$ és $d_2 = \beta$. A fentiekből

$$\mu_1 = \frac{\beta_i - \beta \langle \alpha^i, \alpha \rangle_D}{1 - \langle \alpha^i, \alpha \rangle_D^2} \text{ és } \mu_2 = \frac{\beta - \beta_i \langle \alpha^i, \alpha \rangle_D}{1 - \langle \alpha^i, \alpha \rangle_D^2}. \tag{5.3}$$

Azt állítjuk, hogy $\mu_1, \mu_2 > 0$. Valóban, $\langle z - r^0, x \rangle_D^2 \geq \|z' - r^0\|_D^2$ teljesül minden $x \in \bar{K}$-ra, illetve μ_1 és μ_2 egyértelmű megoldásai a $D(z' - r^0) = \mu_1 \alpha^i + \mu_2 \alpha$ és $\|z' - r^0\|_D^2 = \mu : 1\beta_i + \mu_2 \beta$ egyenleteknek. Legyenek

$$\beta' \coloneqq \|z' - r^0\|_D,\ \alpha' \coloneqq (z' - r^0)/\beta',\ \lambda' \coloneqq (\mu_1 e_i + \mu_2 \lambda)/\beta',$$

így $\lambda' \geq 0$ és $(\alpha', \beta') = \sum_{j=1}^n \lambda'_j (\alpha^j, \beta_j)$. Így z' és λ' a buborék algoritmus minden iterációja során $O(n)$ lépésben kiszámítható.

5.4.4. Állítás. *Mivel $z' \in \{x : \langle \alpha^i, x \rangle_D = \beta_i\}$, így $\|z'\|_D \geq \|r^i\|_D$. Minden iteráció során ha $\lambda_j > 0$, akkor $|\beta_j| \leq \beta$.*

Most tekintsük azt az esetet, amikor $K = \emptyset$.
5.4.3 miatt $\bar{K} = \emptyset$ és az $\alpha^i s \alpha$ vektorok lineárisan összefüggenek. Tehát valamilyen $\nu > 0$ számra $\alpha^i = -\nu\alpha$ és $\beta_i > -\nu\beta$. Ha $\lambda' \coloneqq e_i + \nu\lambda$, akkor $\sum_{j=1}^n \lambda'_j \alpha^j = 0$ és $\sum_{j=1}^n \lambda'_j \beta_j > 0$. Legyenek $v' \coloneqq v^0 + \sum_{j=1}^n \lambda'_j v^j$ és $w' \coloneqq \sum_{j=1}^n \lambda'_j w_j e_j$. Ekkor $A^T v' + w' = 0$, $w' \geq 0$ és $b^T v' + l^T w' > 0$, ami igazolja, hogy a $\tilde{P}$ poliéder üres, mivel a $\tilde{P}$ poliéder duális feladatának egy megoldása.

Függelék

5.5. Jelölések

Egy v vektor esetén $\hat{v}$ jelöli v irányú egységvektort, vagyis $\hat{v} \coloneqq \frac{v}{\|v\|}$. Jelölje e_i az i. egységvektort, $\mathbf{1}$ a csupa 1-est tartalmazó, míg $\mathbf{0}$ a csupa 0-ból álló vektort, ahol a dimenzió feladatonként egyértelmű. $B_D(c,r)$ jelöli a c középpontú, r sugarú gömböt $\mathbb{R}^n$-ben, vagyis $B_D(c,r) \coloneqq \{x \in \mathbb{R}^n : \|c-x\| \leq r\}$.

5.6. Kódolási méret

Az algoritmusok futási idejének meghatározásához szükségünk van egy olyan mennyiség bevezetésére, mely megmutatja, hogy a számítógépen milyen méretű tárterületet foglal egy adott változó (szám, vektor, mátrix). Egész számok esetén bináris kódolást alkalmazunk. Egy $n \neq 0$ egész szám elkódolásánál 1 bit tartalmazza a szám előjelét, $\lceil \log_2(|n|+1) \rceil$ cellányi hely pedig a szám abszolút értékének megfelelő $\{0,1\}$ sorozatot tárolja. A 0 tárolásához csak egy cellára van szükség. Így egy egész szám tárolásához szükséges hely

$$\langle n \rangle \coloneqq 1 + \lceil \log_2(|n|+1) \rceil,\ n \in \mathbb{Z},$$

és $\langle n \rangle$-nek hívjuk az n kódolási, vagy input-méretét. Ha egy algoritmus outputja egy egész szám, akkor az outputon a szám bináris kódját értjük.
Minden racionális szám egyértelműen felírható p/q alakban, ahol $q > 0$, illetve p és q relatív prímek. Egy r racionális szám tárolásához szükséges hely tehát

$$\langle r \rangle \coloneqq \langle p \rangle + \langle q \rangle,$$

és ezt a számot r kódolási méretének hívjuk. Hasonlóan, ha x egy racionális vektor vagy mátrix, akkor x kódolási mérete ($\langle x \rangle$) az őt alkotó komponensek kódolási méreteinek

összege. Egy $a^T x \leq b$ egyenlőtlenségre, ahol $a \in \mathbb{Q}^n$ és $b \in \mathbb{Q}$ a kódolási méret $\langle a \rangle + \langle b \rangle$ és egy egyenlőtlenségrendszer kódolási mérete az őt alkotó egyenlőtlenségek kódolási méreteinek összege.

5.6.1. Lemma. [3] *(a) Bármely r racionális számra $|r| \leq 2^{\langle r \rangle - 1} - 1$*
(b) Bármely $x \in \mathbb{Q}^n$ vektorra $\|x\| \leq \|x\|_1 \leq 2^{\langle x \rangle - n} - 1$.
(c) Bármely $d \in \mathbb{Q}^{n \times n}$ vektorra $|\det D| \leq 2^{\langle D \rangle - n^2} - 1$.

Bizonyítás.
(a) A definícióból közvetlenül következik.
(b) Legyen $x = (x_1, \ldots, x_n)^T$. Az euklideszi normára teljesül, hogy $\|x\| \leq \|x\|_1$. Ekkor (a) miatt

$$1 + \|x\|_1 = 1 + \sum_{i=1}^{n} |x_i| \leq \prod_{i=1}^{n} (1 + |x_i|) \leq \prod_{i=1}^{n} 2^{\langle x_i \rangle - 1} = 2^{\langle x \rangle - n.}$$

(c) bizonyításához jelölje $d_1, \ldots, d_n$ a D sorvektorait.
Az $a_1, \ldots, a_m \in \mathbb{R}^n$ vektorok által kifeszített paralelepipedon a $\mathbf{0}$ és az a_i vektorok konvex burka. Ha A az $n \times m$-es mátrix, melynek oszlopai az a_i vektorok, akkor $\sqrt{\det(A^T A)}$ a szóban forgó paralelepipedon térfogata. Ebből következik a Hadamard-féle egyenlőtlenség,

$$\sqrt{\det(A^T A)} \leq \prod_{i=1}^{n} \|a_i\|$$

mely csak akkor teljesül egyenlőséggel, ha az a_i vektorok egymásra merőlegesek (a paralelepipedon téglatest alakú). Egy $n \times n$-es A mátrix esetén az egyenlőtlenség az alábbi alakban írható fel:

$$|\det A| \leq \prod_{i=1}^{n} \|a_i\|, \tag{5.4}$$

ahol $a_1, \ldots, a_n$ az A oszlopvektorai. Ezt az egyenlőtlenséget D-n alkalmazva és a (b) pontot felhasználva következik, hogy

$$1 + |\det D| \leq 1 + \prod_{i=1}^{n} \|d_i\| \leq \prod_{i=1}^{n} (1 + \|d_i\|) \leq \prod_{i=1}^{n} 2^{\langle d_i \rangle - n} = 2^{\langle D \rangle - n^2}.$$

□ A dolgozatban az A inputmátrix előbbiekben definiált kódolási méretét L-el jelöljük.

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Jüttner Alpárnak, hogy felkeltette az érdeklődésemet a témával kapcsolatban, és lelkesen megválaszolta a felmerülő kérdéseimet, nem csak a szakdolgozat témájában.
Köszönettel tartozom továbbá csoporttársaimnak, barátaimnak. Szerencsésnek érzem magam, hogy egy ilyen inspiráló közösség része lehetek.

2016. 05. 31.

Halasi Valentina

Irodalomjegyzék

[1] G. B. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, 1947.

[2] L.G. Khachiyan, *A polynomial algorithm in linear programming*, 1979

[3] Martin Grötschel, Laszlo Lovasz, Alexander Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 1988

[4] S. Chubanov, *A strongly polynomial algorithm for linear systems having a binary solution*, (Mathematical Programmming 134 (2012), 533-570)

[5] László A. Végh, Giacomo Zambelli, *A polynomial projection-type algorithm for linear programming*, 2014.

[6] Daniel Dadush, László A. Végh,Giacomo Zambelli, *Rescaled coordinate descent methods for Linear Programming*, 2015.

[7] N.K. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica 4 (1984) 373-395.

[8] J. Dunagan, S. Vempala, *A simple polynomial-time rescaling algorithm for solvign linear programs* , Mathematical Programming 114 (2006) 101-114.

[9] G. B. Dantzig, *An "-precise feasible solution to a linear program with a convexity constraint in 1/"2 iterations independent of problem size*, 1992.

[10] M. Epelman and R. M. Freund, *Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system* Mathematical Programming, 88(3) (2000) 451-485.

[11] Sergei Chubanov, *A polynomial projection algorithm for linear programming*, 2013

[12] T. S. Motzkin I. J. Schoenberg, *The relaxation method for linear inequalities*, 1954.

[13] Alexander Schrijver *Theory of linear and integer programming*, 1998.