Sveučilište J.J.Strossmayera u Osijeku
Odjel za matematiku

Rebeka Čordaš

Linearno programiranje i primjene

Diplomski rad

Osijek, 2014.

Sveučilište J.J.Strossmayera u Osijeku
Odjel za matematiku

Rebeka Čordaš

Linearno programiranje i primjene

Diplomski rad

Mentor:
izv.prof.dr.sc. Domagoj Matijević
Komentor:
Ninoslav Čerkez, IN2 d.o.o.

Osijek, 2014.

Sadržaj

1 Uvod

Linearno programiranje je metoda kojom se pokušava postići najbolji ishod (npr. maksimalni profit ili minimalni trošak) u nekom matematičkom modelu čiji su uvjeti iskazani linearnim uvjetima. Početke samog problema linearnog programiranja možemo naći kod Fouriera po kome je jedna od metoda za rješavanje problema linearnog programiranja (Fourier - Motzkinova eliminacija[1]) dobila ime. Tehniku linearnog programiranja je prvi razvio Leonid Kantorovich 1939. godine kako bi za vrijeme II. svjetskog rata planirao troškove i zaradu i na taj način smanjio troškove vojske i povećao gubitke neprijatelja. Ta tehnika nije bila dostupna široj javnosti i zbog toga nije bila korištena u rješavanju svakodnevnih problema sve do 1947. godine kada je George B. Dantzig objavio simpleks metodu, a John von Neumann razvio teoriju dualnosti kao rješenje problema linearnog programiranja i primijenio ju u području teorije igara. Nakon rata, mnoge su industrije pronašle korist linearne optimizacije (u koju se ubrajaju i tehnike rješavanja problema pomoću linearnog programiranja) u svakodnevnim planiranjima troškova i zarade.

Linearno programiranje je vrlo važno polje u optimizaciji. Mnogi praktični problemi u operacijskim istraživanjima se mogu iskazati kao problemi linearnog programiranja. Kroz povijest su ideje iz područja linearnog programiranja pridonijele razvitku glavnih koncepata teorije optimizacije kao što su dualnost, dekompozicija i važnost konveksnosti i njenih generalizacija. Isto tako, linearno programiranje se koristi u mikroekonomiji i upravljanju tvrtkama u planiranju, proizvodnji, tehnologiji, prijevozu i ostalim problemima. Iako se problemi vezani za upravljanje tvrtkama neprestano mijenjaju, većina tvrtki ima u cilju maksimizirati profit i minimizirati troškove uz ograničene resurse, pa se većina problema može karakterizirati kao problem linearnog programiranja i riješiti uz neku od poznatih i prihvaćenih metoda.

Pogledajmo jedan svakodnevni problem koji se može riješiti vrlo lako ako ga modeliramo kao problem linearnog programiranja (iz [3]).

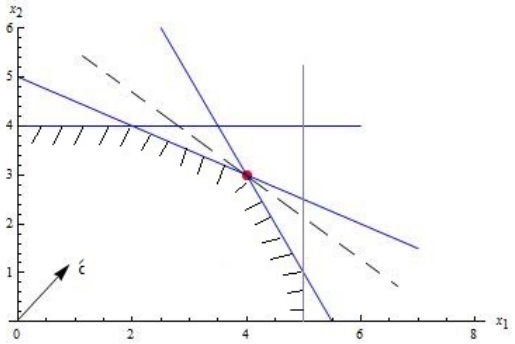
Primjer 1.1. Uzmimo jednog proizvođača negaziranih pića. On pravi dvije vrste soka: Spring i Nebsi od sastojaka A i B te vode. Da bi se napravilo $100l$ Springa potrebno je $3l$ sastojka A i $8l$ sastojka B, a za Nebsi je potrebno $6l$ sastojka A i $4l$ sastojka B. Sto litara Springa nosi zaradu od $100kn$, a $100l$ Nebsija zaradu od $125kn$. Proizvođač u skladištu ima na raspolaganju $30l$ sastojka A i $44l$ sastojka B. Napravljene sokove sipa u bačve. Za Spring ima bačvu u koju stane $500l$ tekućine, a za Nebsi bačvu kapaciteta $400l$. Na kraju dana dolazi preprodavač koji kupuje sok. Cilj proizvođača sokova je napraviti koliko god može soka, uzimajući u obzir dana ograničenja, kako bi maksimizirao svoj profit. Drugim riječima, treba napraviti plan proizvodnje koji će maksimizirati profit. Uz pretpostavku da je $(x_1, x_2) \in \mathbb{R}^2$ traženi plan proizvodnje, ovaj problem se može prikazati kao problem

[1]Metoda rješavanja problema linearnog programiranja eliminacijom varijabli iz sustava linearnih nejednadžbi. Za više informacija vidi: `http://en.wikipedia.org/wiki/Fourier-Motzkin_elimination`

linearnog programiranja na sljedeći način:

$$\begin{aligned} 100x_1 + 125x_2 &\to \max \\ 3x_1 + 6x_2 &\leq 30 \\ 8x_1 + 4x_2 &\leq 44 \\ x_1 &\leq 5 \\ x_2 &\leq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Kako imamo dvije varijable u ovom problemu, možemo ga riješiti grafički. Na slici (1) su ucrtani uvjeti (plavi pravci) i vektor $c = (100, 125)^T$ (pomnožen sa skalarom $1/100$) koji je pomnožen sa vektorom $(x_1, x_2)^T$ u funkciji koju je potrebno maksimizirati. Točka $(4, 3)$



Slika 1: Grafički prikaz problema

je označena crvenom bojom i ona predstavlja optimalno rješenje zadanog problema koje smo dobili tako da smo pravac okomit na vektor c (iscrtkani pravac sa slike (1)) paralelno pomicali u smjeru vektora c sve dok nismo došli do one točke u kojoj napuštamo područje određeno zadanim uvjetima. Kretali smo se u smjeru vektora c jer se kretanjem u tom smjeru vrijednost funkcije $100x_1 + 125x_2$ povećava, a cilj nam je maksimizirati profit. U slučajevima kada je potrebno minimizirati neku funkciju, krećemo se u smjeru suprotnom od smjera vektora c.

Kao što smo mogli vidjeti iz prethodnog primjera, ako imamo problem sa samo dvije varijable, lako ga je riješiti koristeći grafički prikaz. No, ako imamo problem sa više od tri varijable, potrebno je pronaći drugi način za rješavanje, pa su se iz tog razloga razvile brojne metode za rješavanje problema linearnog programiranja. Geometrija lineranog programiranja, kojom se bavimo u 3. poglavlju ovog rada, nam olakšava da si na neki način predočimo koje je to područje u kojem se nalaze točke koje zadovoljavaju uvjete problema. Jedna od metoda za rješavanje problema linearnog programiranja je i simpleks metoda koja je tema 4. poglavlja ovog rada.

2 Problem linearnog programiranja

U uvodnom primjeru smo vidjeli kako izgleda jedan problem linearnog programiranja. Sada ćemo definirati sam problem linearnog programiranja, funkciju cilja, dopustivo rješenje i dopustivo područje te optimalno rješenje problema linearnog programiranja.

Definicija 2.1. Neka su $c \in \mathbb{R}^n$, $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $i \in M \subset \mathbb{N}$ te neka je $f : \mathbb{R}^n \to \mathbb{R}$ definirana sa $f(x) = c^T x$. Promotrimo sljedeći optimizacijski problem:

$$f(x) = c^T x \to \min \tag{1}$$

uz uvjete

$$a_i^T x \geq b_i, \ \ i \in M_1 \tag{2}$$
$$a_i^T x \leq b_i, \ \ i \in M_2 \tag{3}$$
$$a_i^T x = b_i, \ \ i \in M_3 \tag{4}$$

gdje je $M_1 \cup M_2 \cup M_3 = M$ i $M_i \cap M_j = \emptyset$, $i \neq j$. Problem (1) - (4) zovemo **LP problem** (problem linearnog programiranja). Pri tome funkciju f zovemo **funkcija cilja**. Vektor $x \in \mathbb{R}^n$ sa svojstvima (2) - (4) zovemo **dopustivo rješenje**. Skup svih dopustivih rješenja zovemo **dopustivo područje**. Za dopustivo rješenje x^* kažemo da je **optimalno dopustivo rješenje** ako vrijedi

$$f(x^*) = c^T x^* \leq c^T x = f(x), \ \ \forall \text{ dopustivi } x$$

Uočimo da vrijedi:

- maksimizacijski problem $c^T x \to \max$ se može svesti na minimizacijski problem $-c^T x \to \min$
- $a_i^T x = b_i \iff a_i^T x \leq b_i \ \& \ a_i^T x \geq b_i$
- $a_i^T x \leq b_i \iff -a_i^T x \geq -b_i$

Imajući to na umu, možemo zaključiti da se svaki LP problem može zapisati na sljedeći način:

$$c^T x \to \min$$

uz uvjet $Ax \geq b$, gdje je $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ i $c \in \mathbb{R}^n$. Zapišimo primjer 1.1 iz uvoda na ovaj način:

Primjer 2.1.

$$\begin{aligned} -100x_1 - 125x_2 &\to \min \\ -3x_1 - 6x_2 &\geq -30 \\ -8x_1 - 4x_2 &\geq -44 \\ -x_1 &\geq -5 \\ -x_2 &\geq -4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Iz ovog problema možemo iščitati da je $c = [-100, -125]^T$, $b = [-30, -44, -5, -4, 0, 0]^T$ te

$$A^T = \begin{bmatrix} -3 & -8 & -1 & 0 & 1 & 0 \\ -6 & -4 & 0 & -1 & 0 & 1 \end{bmatrix}$$

pa problem uistinu ima oblik $c^T x \to \min$ uz uvjet $Ax \geq b$.

Napomena. Ako su $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, onda se minimizacijski problem $c^T x \to \min$ uz uvjete $Ax = b$, $x \geq 0$ zove **standardni oblik problema linearnog programiranja**.

Propozicija 2.1. *Svaki LP problem se može zapisati kao standardni oblik problema linearnog programiranja.*

2.1 Neki problemi koji se svode na LP probleme

U ovom poglavlju navest će se nekoliko primjera problema koji se mogu svesti na problem linearnog programiranja.

Primjer 2.2. ***Problem proizvodnje***
Neka tvrtka proizvodi n različitih dobara koristeći m različitih sirovina. Neka je s b_i, $i = 1, \ldots, m$ označena dostupna količina i - te sirovine. Za proizvodnju j - tog dobra, $j = 1, \ldots, n$, koje donosi zaradu c_j po jedinici, potrebno je a_{ij} jedinica i - tog materijala. Tvrtka mora odlučiti koliko će svakog dobra proizvesti kako bi maksimizirala svoju zaradu.

Označimo sa $x_j, j = 1, \ldots n$ količinu j - tog dobra. Tada navedeni problem možemo zapisati na sljedeći način:

$$c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \to \max$$

uz uvjete

$$\begin{aligned} a_{i1} x_1 + \cdots + a_{in} x_n \leq b_i, \qquad & i = 1, \ldots, m, \\ x_j \geq 0, \qquad & j = 1, \ldots, n. \end{aligned}$$

Primjer 2.3. ***Problem optimalne prehrane***
Pretpostavimo da imamo na raspolaganju namirnice $N_1, \ldots, N_n$. Cijena po jedinici namirnice je c_j, $j = 1, \ldots, n$. U namirnicama su prisutni nutritivni elementi $E_1, \ldots, E_m$, pri čemu u namirnici N_j ima a_{ij} nutritivnog elementa E_i, $j = 1, \ldots, n$, $j = 1, \ldots, m$. Svaka osoba tijekom dana mora unijeti barem b_i jedinica nutritivnog elementa E_i. Potrebno je formulirati sljedeći problem: Koliko treba konzumirati pojedine namirnice da bi se zadovoljila dnevna potreba za nutritivnim elementima, a da se pri tome minimizira cijena prehrane?

Ako sa x_j, $j = 1, \ldots, n$, označimo količinu konzumirane namirnice N_j, navedeni problem možemo zapisati na sljedeći način:

$$c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \to \min$$

uz uvjete

$$\begin{gathered} a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n \geq b_2 \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_m \\ x_1, x_2, \ldots, x_m \geq 0 \end{gathered}$$

tj. u matričnom obliku:

$$\begin{gathered} c^T x \to \min \\ Ax \geq b \\ x \geq 0 \end{gathered}$$

Primjer 2.4. ***Problem najboljeg pravca***
Zadani su podaci (t_i, y_i), $i = 1, \ldots, r$. Potrebno je odrediti pravac s jednadžbom $y = kx + l$, $k, l \in \mathbb{R}$ koji u l_1 smislu najbolje aproksimira dane podatke.

Želimo $F_1(k, l) = \sum_{i=1}^{r} |kt_i + l - y_i| \to \min_{(k,l)}$. Označimo

$$z_i := |kt_i + l - y_i| = \max\{kt_i + l - y_i, -kt_i - l + y_i\}, \qquad i = 1, \ldots, r$$

Sada navedeni problem možemo zapisati u obliku

$$c^T x \to \min \qquad \text{uz uvjet} \qquad Ax \leq b$$

gdje su

$$\begin{gathered} c = [1, 1, 1, \ldots, 1, 0, 0] \in \mathbb{R}^{r+2} \\ x = [z_1, z_2, \ldots, z_r, k, l] \in \mathbb{R}^{r+2} \\ b = [y_1, -y_1, y_2, -y_2, \ldots, y_r, -y_r] \in \mathbb{R}^{2r} \\ A = \begin{bmatrix} -1 & 0 & \ldots & 0 & t_1 & 1 \\ -1 & 0 & \ldots & 0 & -t_1 & -1 \\ 0 & -1 & \ldots & 0 & t_2 & 1 \\ 0 & -1 & \ldots & 0 & -t_2 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & -1 & t_r & 1 \\ 0 & 0 & \ldots & -1 & -t_r & -1 \end{bmatrix} \in \mathbb{R}^{(2r)\times(r+2)} \end{gathered}$$

Primjer 2.5. ***Problem rasporeda***
Bolnica želi napraviti tjedne noćne smjene za svoje medicinske sestre. Potražnja za medicinskim sestrama za noćnu smjenu za j - ti dan je cijeli broj d_j, $j = 1, \ldots, 7$. Svaka medicinska sestra radi pet dana zaredom u noćnoj smjeni. Potrebno je pronaći minimalan broj medicinskih sestara koje bolnica treba zaposliti da bi se mogle napraviti takve smjene.

Ako varijablom y_j označimo broj medicinskih sestara koje rade na j - ti dan, nećemo biti u mogućnosti obuhvatiti uvjet da svaka medicinska sestra treba raditi 5 dana zaredom. Zato ćemo varijable odabrati drukčije i sa x_j označiti broj medicinskih sestara koje svoj radni tjedan započinju j - ti dan. Sada problem možemo zapisati na sljedeći način:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \to \min$$

uz uvjete

$$\begin{aligned}
x_1 + x_4 + x_5 + x_6 + x_7 &\geq d_1 \\
x_1 + x_2 + x_5 + x_6 + x_7 &\geq d_2 \\
x_1 + x_2 + x_3 + x_6 + x_7 &\geq d_3 \\
x_1 + x_2 + x_3 + x_4 + x_7 &\geq d_4 \\
x_1 + x_2 + x_3 + x_4 + x_5 &\geq d_5 \\
x_2 + x_3 + x_4 + x_5 + x_6 &\geq d_6 \\
x_3 + x_4 + x_5 + x_6 + x_7 &\geq d_7 \\
x_j \geq 0, \qquad x_j \in \mathbb{Z}, \qquad & j = 1, \ldots, 7
\end{aligned}$$

Ovo bi bio problem linearnog programiranja da nema dodatnog uvjeta da je $x_j \in \mathbb{Z}$. Zbog toga je ovo zapravo problem cjelobrojnog linearnog programiranja. Jedan način na koji se može riješiti ovakav problem je ignoriranjem uvjeta cjelobrojnosti. Na taj način dobijemo relaksaciju problema linearnog programiranja. Kako taj problem ima manje uvjeta od početnog, njegova optimalna vrijednost funkcije cilja će biti manja ili jednaka optimalnoj vrijednosti funkcije cilja početnog problema. Ako je optimalno rješenje relaksacije cjelobrojno, onda je to ujedno i rješenje početnog problema. U suprotnom, svaki x_j zaokružimo na sljedeći cijeli broj veći od x_j i tako dobijemo dopustivo, ali ne nužno i optimalno rješenje cjelobrojnog problema linearnog programiranja. Problem iz ovog primjera je lako rješiv, ali u pravilu su problemi cjelobrojnog linearnog programiranja vrlo teško rješivi problemi.

3 Geometrija linearnog programiranja

U prethodnom poglavlju se govorilo o problemu linearnog programiranja. U ovom poglavlju ćemo doći do jednog od načina rješavanja ovog problema. Pokazat ćemo da ako je problem linearnog programiranja dopustiv i omeđen i postoje vrhovi, postoji i optimalno rješenje tog LP problema i to rješenje je upravo jedan od vrhova.

3.1 Osnovni pojmovi

Kako bismo mogli pričati o geometriji linearnog programiranja, potrebno je uvesti neke osnovne pojmove.

Definicija 3.1. Za skup $S \subseteq \mathbb{R}^n$ kažemo da je **konveksan** ako $\forall x, y \in S, \forall \lambda \in [0,1]$ vrijedi $\lambda x + (1-\lambda)y \in S$.

Definicija 3.2. Za funkciju f kažemo da je **konveksna** na D ako $\forall x, y \in D$ i $\forall \lambda \in [0,1]$ vrijedi

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y).$$

Definicija 3.3. Neka su $x_i \in \mathbb{R}^n$ i $\lambda_i \in [0,1]$ za $i \in 1, \ldots, k$ te $\sum_{i=1}^{k} \lambda_i = 1$. Vektor $x = \sum_{i=1}^{k} \lambda_i x_i$ zovemo **konveksna kombinacija** vektora x_i.

Sljedeći teorem nam govori da konveksne funkcije na konveksnim skupovima postižu globalne ekstreme:

Teorem 3.1. *Neka je $f : \mathcal{P} \to \mathbb{R}$ konveksna funkcija definirana na konveksnom skupu $\mathcal{P} \subseteq \mathbb{R}^n$. Ako funkcija f u $x^* \in \mathcal{P}$ postiže lokalni minimum, onda f u x^* postiže i svoj globalni minimum.*

Dokaz. Neka je $x^* \in \mathcal{P}$ točka lokalnog minimuma funkcije f. To znači da postoji $r > 0$ takav da vrijedi $f(x^*) < f(y)$, $\forall y \in K(x^*, r) \cap \mathcal{P}$. Uzmimo proizvoljan vektor $x \neq x^* \in \mathcal{P}$ i definirajmo vektor $y(\lambda) = \lambda x + (1-\lambda)x^*$, $\lambda \in [0,1]$. Kako je $\mathcal{P}$ konveksan skup, slijedi da je $y \in \mathcal{P}$ kao konveksna kombinacija dvaju vektora iz $\mathcal{P}$. Definirajmo broj $\lambda_0 < \min\{1, r\|x - x^*\|^{-1}\}, \lambda_0 \in \langle 0, 1]$. Tada za svaki $\lambda \in \langle 0, \lambda_0]$ vrijedi

$$\|y(\lambda) - x^*\| = \|\lambda x + (1-\lambda)x^* - x^*\| = \lambda\|x - x^*\| \leq \lambda_0\|x - x^*\| < r$$

Sada znamo da je $y(\lambda) \in K(x^*, r) \cap \mathcal{P}$ pa vrijedi

$$\begin{aligned} f(x^*) \leq f(y(\lambda)) = f(\lambda x + (1-\lambda)x^*) &\leq \lambda f(x) + (1-\lambda)f(x^*) \\ \lambda f(x^*) &\leq \lambda f(x) / : \lambda > 0 \\ f(x^*) &\leq f(x) \end{aligned}$$

Kako je $x \in \mathcal{P}$ proizvoljan, pokazali smo da je x^* točka globalnog minimuma funkcije f. □

U sljedeće dvije definicije definiramo pojmove poliedra, hiperravnine i poluprostora.

Definicija 3.4. Neka su $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$. Skup $\{x \in \mathbb{R}^n : Ax \geq b\} \subseteq \mathbb{R}^n$ zovemo **poliedar** u $\mathbb{R}^n$.

Definicija 3.5. Neka su $a \in \mathbb{R}^n$, $a \neq 0$, $b \in \mathbb{R}$. Skup $\{x \in \mathbb{R}^n : a^T x = b\}$ zovemo **hiperravnina** s vektorom normalne a, a skup $\{x \in \mathbb{R}^n : a^T x \geq b\}$ zovemo **poluprostor**.

Propozicija 3.1 govori da je poliedar konveksan skup. To je važno jer se uvjeti problema linearnog programiranja mogu zapisati kao poliedar, a ako je on konveksan skup, znamo da mu možemo naći globalni minimum (maksimum), a to znači da možemo naći i optimalno rješenje zadanog problema linearnog programiranja.

Propozicija 3.1. 1) *Poluprostor u $\mathbb{R}^n$ je konveksan skup.*

2) *Presjek konačno mnogo konveksnih skupova je konveksan skup.*

3) *Poliedar u $\mathbb{R}^n$ je konveksan skup.*

Dokaz. 1) Neka je $\Pi = \{x \in \mathbb{R}^n : a^T x \geq b\}$ dani poluprostor, te neka su $x, y \in \Pi$ i $\lambda \in [0, 1]$. Pogledajmo sljedeću nejednakost:

$$a^T(\lambda x + (1-\lambda)y) = \lambda a^T x + (1-\lambda)a^T y \geq \lambda b + (1-\lambda)b = b.$$

Vidimo da je $\lambda x + (1-\lambda)y \in \Pi$, pa je Π konveksan skup.
2) Dokažimo navedenu tvrdnju za dva konveksna skupa S_1 i S_2. Pretpostavimo da je $S_1 \cap S_2 \neq \emptyset$. Neka su $x, y \in S_1 \cap S_2$. Iz toga znamo da su $x, y \in S_1$ i $x, y \in S_2$. Kako su S_1 i S_2 konveksni, za svaki $\lambda \in [0, 1]$ vrijedi $\lambda x + (1-\lambda)y \in S_1$ i $\lambda x + (1-\lambda)y \in S_2$, a to znači da je $\lambda x + (1-\lambda)y \in S_1 \cap S_2$ pa je $S_1 \cap S_2$ konveksan skup. Metodom matematičke indukcije se može pokazati da je presjek konačno mnogo konveksnih skupova konveksan skup.
3) Poliedar je presjek konačno mnogo poluprostora, pa tvrdnja slijedi iz 1) i 2). □

3.2 Ekstremne točke, vrhovi i bazična dopustiva rješenja

Već je prije spomenuto da je važno to što je poliedar konveksan skup jer to znači da on ima globalni minimum, pa možemo pronaći i rješenje pripadnog LP problema. Globalni minumum se u poliedru postiže u jednom od vrhova poliedra. U ovom ćemo potpoglavlju definirati što je to vrh poliedra, ekstremna točka i bazično dopustivo rješenje i pokazati da su ta tri pojma ekvivalentna.

Definicija 3.6. Neka je $\mathcal{P}$ poliedar u $\mathbb{R}^n$. Vektor $x \in \mathcal{P}$ je **ekstremna točka** poliedra $\mathcal{P}$ ako ne postoje vektori $y, z \in \mathcal{P}$, $y \neq x, z \neq x$ i skalar $\lambda \in [0, 1]$ takvi da je $x = \lambda y + (1-\lambda)z$.

Geometrijski to znači da je ekstremna točka poliedra ona točka koja se ne može napisati kao netrivijalna konveksna kombinacija točaka iz poliedra.

Definicija 3.7. Neka je $\mathcal{P}$ poliedar u $\mathbb{R}^n$. Vektor $x \in \mathcal{P}$ je **vrh** poliedra ako postoji vektor $c \in \mathbb{R}^n$ takav da za svaki $y \in \mathcal{P}$, $y \neq x$ vrijedi $c^T x < c^T y$.

Geometrijski to znači da je vrh poliedra ona točka poliedra kroz koju možemo provući hiperravninu sa svojstvom da se sve točke poliedra (izuzev vrha) nalaze s iste strane te poluravnine.

Definicija 3.8. Neka je poliedar $\mathcal{P} \subseteq \mathbb{R}^n$ zadan na sljedeći način:

$$\begin{aligned} a_i^T x &\geq b_i, \quad i \in M_1 \\ a_i^T x &\leq b_i, \quad i \in M_2 \\ a_i^T x &= b_i, \quad i \in M_3 \end{aligned}$$

Ako za vektor x^* vrijedi $a_{i_0}^T x^* = b_i$, za neki $i_0 \in M_1 \cup M_2 \cup M_3$, onda kažemo da je uvjet s indeksom i_0 **aktivan** u x^*.

Definicija 3.9. Neka je poliedar $\mathcal{P} \in \mathbb{R}^n$ zadan kao u definiciji 3.8 te neka je $I = \{i \in M_1 \cup M_2 \cup M_3 : a_i^T x^* = b_i\}$ skup indeksa aktivnih u vektoru $x^* \in \mathbb{R}^n$. Za vektor $x^* \in \mathbb{R}^n$ kažemo da je **bazično rješenje** ako su svi uvjeti koji sadrže jednakost aktivni u x^* te ako skup $\{a_i, i \in I\}$ sadrži n linearno nezavisnih vektora. Bazično rješenje $x^* \in \mathbb{R}^n$ je **bazično dopustivo rješenje** ako zadovoljava sve uvjete problema linearnog programiranja.

Teorem 3.2. *Neka je $\mathcal{P} \in \mathbb{R}^n$ poliedar i neka je $x^* \in \mathcal{P}$. Sljedeće tri tvrdnje su ekvivalentne:*

1. *x^* je vrh poliedra,*
2. *x^* je ekstremna točka poliedra,*
3. *x^* je bazično dopustivo rješenje.*

Dokaz. Dokaz ćemo provesti u tri koraka. Pokazat ćemo da 1) $\Rightarrow$ 2), 2) $\Rightarrow$ 3) i na posljetku 3) $\Rightarrow$ 1).
1) $\Rightarrow$ 2) Neka je x^* vrh poliedra. To znači da postoji $c \in \mathbb{R}^n$ takav da vrijedi

$$c^T x^* < c^T x, \quad \forall x \in \mathcal{P}, \quad x \neq x^*. \tag{5}$$

Pretpostavimo da x^* nije ekstremna točka, tj. da postoje vektori $y, z \in \mathcal{P}$, $y \neq x^*$, $z \neq x^*$, te $\lambda \in [0, 1]$ takvi da je

$$x^* = \lambda y + (1 - \lambda) z \tag{6}$$

Iz (5) znamo da je $c^T x^* < c^T y$ i $c^T x^* < c^T z$. Sada imamo:

$$\begin{aligned} c^T x^* &= \lambda c^T x^* + (1 - \lambda) c^T x^* \\ &< \lambda c^T y + (1 - \lambda) c^T z \\ &= c^T (\lambda y + (1 - \lambda) z) \end{aligned}$$

A to je u kontradikciji s pretpostavkom (6).

2) $\Rightarrow$ 3) Ovu implikaciju ćemo dokazati kontrapozicijom. Pretpostavit ćemo da x^* nije BDR i pokazat ćemo da tada x^* nije ni ekstremna točka. Pretpostavimo da je poliedar $\mathcal{P}$ zapisan kao:

$$a_i^T x \geq b_i, \quad i \in M_1$$
$$a_i^T x = b_i, \quad i \in M_2$$

Pretpostavimo da x^* nije BDR. To znači da je broj linearno nezavisnih uvjeta koji su aktivni u x^* manji od n. Ako sa I označimo skup indeksa aktivnih u x^*, skup $\{a_i, i \in I\} \subseteq \mathbb{R}^n$ je pravi potprostor od $\mathbb{R}^n$, pa postoji $d \in \mathbb{R}^n \backslash \{0\}$ takav da vrijedi $a_i^T d = 0$, $\forall i \in I$. Definirajmo vektore $y = x^* + \epsilon d$, $z = x^* - \epsilon d$ ($\epsilon > 0$) i pokažimo da su $y, z \in \mathcal{P}$. Ako je $i \in I$ onda vrijedi

$$a_i^T y = a_i^T (x^* + \epsilon d) = a_i^T x^* + \epsilon a_i^T d = a_i^T x^* = b_i$$

Ako $i \notin I$, pokažimo da tada vrijedi $a_i^T x^* > b_i$, pa će to značiti da je $y \in \mathcal{P}$. Vrijedi $a_i^T y = a_i^T (x^* + \epsilon d) = a_i^T x^* + \epsilon a_i^T d$. Ako je $a_i^T d \geq 0$ slijedi da je $a_i^T x^* > b_i$, pa smo pokazali da je $y \in \mathcal{P}$. Ako je $a_i^T d < 0$, možemo odabrati dovoljno mali $\epsilon > 0$ za koji će vrijediti $a_i^T x^* + \epsilon a_i^T d > b_i$. Pogledajmo koliki može biti ϵ.

$$\epsilon a_i^T d > b_i - a_i^T x^* \quad / : a_i^T d$$
$$\epsilon < \frac{b_i - a_i^T x^*}{a_i^T x^*}$$

Za svaki $\epsilon \in \langle 0, \frac{b_i - a_i^T x^*}{a_i^T x^*} \rangle$ vrijedi $a_i^T y > b_i$. Analogno se pokaže da je $z \in \mathcal{P}$. Sada vrijedi

$$x^* = \frac{1}{2}(y + z) = \frac{1}{2}(x^* + \epsilon d + x^* - \epsilon d) = x^*$$

pa možemo zaključiti da x^* nije ekstremna točka.

3) $\Rightarrow$ 1) Neka je x^* BDR i $I = \{i : a_i^T x^* = b_i\}$ skup aktivnih indeksa u x^*. Definirajmo vektor $c = \sum_{i \in I} a_i$. Tada je $c^T x^* = \sum_{i \in I} a_i^T x^* = \sum_{i \in I} b_i$. Za proizvoljan $x \in \mathcal{P}$ imamo

$$c^T x = \sum_{i \in I} a_i^T x \geq c^T x^*. \tag{7}$$

Iz toga znamo da je x^* rješenje LP problema $c^T x \to \min$ uz uvjet $x \in \mathcal{P}$. Pri tome jednakost u (7) vrijedi onda i samo onda ako je $a_i^T x = b_i$, $\forall i \in I$. Kako je x^* BDR, broj linearno nezavisnih vektora a_i, $i \in I$ aktivnih u x^* je jednak n, pa sustav $a_i^T x = b_i$, $i \in I$ ima jedinstveno rješenje, a to je upravo x^*. Pokazali smo da postoji vektor $c \in \mathbb{R}^n$ takav da vrijedi $c^T x^* \leq c^T x$, $\forall x \neq x^*$, $x \in \mathcal{P}$, tj. da je x^* vrh poliedra $\mathcal{P}$. $\square$

3.3 Konstrukcija BDR-a, degenerativnost i egzistencija ekstremne točke

U prethodnom potpoglavlju smo pokazali da je su pojmovi BDR, ekstremna točka i vrh poliedra ekvivalentni. U ovom potpoglavlju ćemo vidjeti koji su uvjeti za postojanje BDR-a i ekstremne točke poliedra. Dat ćemo postupak za konstrukciju BDR-a i definirati degenerativno rješenje.

Teorem 3.3. *Neka je* $\mathcal{P} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ *te* $rang(A) = m$, $m < n$. *Vektor* $x \in \mathbb{R}^n$ *je bazično rješenje onda i samo onda ako je* $Ax = b$, *te ako postoje indeksi* $B(1), B(2), \ldots, B(m)$ *takvi da vrijedi:*

- *stupci* $A_{B(1)}, \ldots, A_{B(m)}$ *su linearno nezavisni*
- *ako* $i \notin \{B(1), \ldots, B(m)\}$, *onda je* $x_i = 0$.

Sada možemo napisati postupak za konstrukciju bazičnog rješenja:

1. Odabrati m linearno nezavisnih stupaca $A_{B(1)}, \ldots, A_{B(m)}$ matrice A.
2. Staviti $x_i = 0$, $\forall i \notin \{B(1), \ldots, B(m)\}$.
3. Riješiti sustav $Bx_B = b$, gdje je $B = [A_{B(1)}, \ldots, A_{B(m)}]$ i $x_B = [x_{B(1)}, \ldots, x_{B(m)}]^T$.
4. Vektor $x = [x_1, x_2, \ldots, x_n]^T$ takav da je

$$x_j = \begin{cases} 0 & j \notin \{B(1), \ldots, B(m)\} \\ x_{B(i)} & j = B(i) \end{cases}$$

je bazično rješenje.

Varijable $x_{B(1)}, \ldots, x_{B(m)}$ zovemo **bazične varijable**, vektore $A_{B(1)}, \ldots, A_{B(m)}$ **bazični vektori**, indekse $B(1), \ldots, B(m)$ **bazični indeksi** a matricu B **matrica baze**.

Sada ćemo definirati degenerativno rješenje i degenerativno bazično rješenje i na primjeru pokazati kako se pronalazi bazično rješenje te kako izgleda degenerativno bazično rješenje.

Definicija 3.10. Neka je $\mathcal{P} = \{x \in \mathbb{R}^n : a_i^T x \geq b, i = 1, \ldots m\}$. Za bazično rješenje $x^* \in \mathbb{R}^n$ kažemo da je **degenerativno** ako je više od n uvjeta aktivno u x^*.

Definicija 3.11. Neka je $\mathcal{P} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ poliedar u standardnom obliku te neka je x bazično rješenje. Vektor x je **degenerativno bazično rješenje** ako je više od $n - m$ komponenti od x jednako 0.

Primjer 3.1. Zadan je poliedar $\mathcal{P} = \{[x_1, x_2, x_3]^T : x_1 - x_2 = 0, x_1 + x_2 + 2x_3 = 2, x_1, x_2, x_3 \geq 0\}$. Treba pronaći bazična dopustiva rješenja i odrediti jesu li rješenja degenerativna.

Iz zapisa poliedra lako možemo iščitati da je $A = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 2 \end{bmatrix}$ i $b = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$. Prateći postupak za konstrukciju bazičnog rješenja, prvo trebamo pronaći dva linearno nezavisna stupca matrice A (jer je $m = 2$) i formirati matricu baze B. Uzmimo prva dva stupca matrice A. Time smo dobili matricu baze $B = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$. Kako smo matricu B napravili od prva dva stupca matrice A, postavimo $x_3 = 0$. Sada trebamo izračunati x_B iz sustava $Bx_B = b$. Jednostavnim računom se dobije da je $x_B = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, tj. da je $x^* = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$. Kako je x^* aktivan u uvjetima $x_1 - x_2 = 0$, $x_1 + x_2 + 2x_3 = 2$ i $x_3 = 0$, možemo zaključiti da je x^* nedegenerativno bazično dopustivo rješenje. Isto tako, mogli smo za matricu B odabrati 2. i 3. stupac matrice A ili 1. i 3. stupac matrice A. U tim bi slučajevima za x^* dobili vektor $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ koji je aktivan u četiri uvjeta, pa je to rješenje degenerativno.

Sljedeća definicija i teoremi govore o egzistenciji optimalnog rješenja problema linearnog programiranja.

Definicija 3.12. Kažemo da poliedar $\mathcal{P} \subseteq \mathbb{R}^n$ sadrži pravac ako postoji vektor $x \in \mathcal{P}$ te vektor $d \in \mathbb{R}^n \backslash \{0\}$ takav da je $x + \lambda d \in \mathcal{P}, \forall \lambda \in \mathbb{R}$.

Teorem 3.4. *Neka je* $\mathcal{P} = \{x \in \mathbb{R}^n : a_i^T x \geq b, i = 1, \dots m\} \neq \emptyset$. *Onda su sljedeće tvrdnje ekvivalentne:*

- *Poliedar* $\mathcal{P}$ *ima barem jednu ekstremnu točku.*
- *Poliedar ne sadrži pravac.*
- *Među vektorima* $a_1, \dots, a_m$ *ima* n *linearno nezavisnih.*

Teorem 3.5. *Zadan je LP problem* $c^T x \to \min$, $x \in \mathcal{P}$. *Pretpostavimo da* $\mathcal{P}$ *ima barem jednu ekstremnu točku te da LP problem ima optimalno rješenje. Onda postoji optimalno rješenje koje je ekstremna točka poliedra* $\mathcal{P}$.

Dokaz. Neka je Q skup svih optimalnih rješenja LP problema, $Q \neq \emptyset$. Neka je $\mathcal{P}$ poliedar oblika $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \geq b\}$. Pretpostavimo da je optimalna vrijednost funkcije cilja jednaka v. $Q = \{x \in \mathbb{R}^n : Ax \geq b, c^T x = v\}$ je poliedar i $Q \subset \mathcal{P}$. Kako $\mathcal{P}$ ima barem jednu ekstremnu točku, iz prethodnog teorema znamo da $\mathcal{P}$ ne sadrži pravac. To znači da Q ima barem jednu ekstremnu točku. Označimo ju sa x^*. Pokazat ćemo da je x^* ekstremna točka od $\mathcal{P}$. Pretpostavimo suprotno, tj. da x^* nije ekstremna točka od $\mathcal{P}$. To znači da postoje $y, z \in \mathcal{P}$, $y \neq x^*$, $z \neq x^*$ i $\lambda \in [0, 1]$ takvi da je $x^* = \lambda y + (1 - \lambda) z$. Kako je $x^* \in Q$ slijedi $v = c^T x^* = c^T(\lambda y + (1 - \lambda) z) = \lambda c^T y + (1 - \lambda) c^T z$. Znamo da je v najmanja moguća vrijednost funkcije f, pa vrijedi $v \leq c^T y$ i $v \leq c^T z$. Iz toga slijedi $v = c^T x^* = \lambda c^T y + (1 - \lambda) c^T z \geq \lambda v + (1 - \lambda) v = v$, pa vrijedi $c^T y = v$ i $c^T z = v$, tj. $y, z \in Q$. x^* smo zapisali kao konveksnu kombinaciju dvaju elemenata iz Q, a to znači da x^* nije ekstremna točka od Q, što je u kontradikciji s pretpostavkom, pa slijedi da je x^* ekstremna točka od $\mathcal{P}$. □

Naposljetku, sljedeći teorem govori o vezi ekstremne točke poliedra i optimalnog rješenja LP problema.

Teorem 3.6. *Zadan je LP problem* $c^T x \to \min$, $x \in \mathcal{P}$. *Pretpostavimo da* $\mathcal{P}$ *ima barem jednu ekstremnu točku. Onda je vrijednost funkcije cilja jednaka* $-\infty$ *ili postoji ekstremna točka koja je optimalna.*

4 Simpleks metoda

Simpleks metoda je jedna od metoda za rješavanje problema linearnog programiranja. Simpleks metoda traži rješenje LP problema "hodajući" po bridovima poliedra koji predstavlja dopustivo područje od jednog bazičnog rješenja do drugog uvijek u smjeru smanjenja troška funkcije cilja. Nakon konačno mnogo iteracija će dostići ono bazično dopustivo rešenje iz kojeg niti jedan brid ne vodi smanjenju troška funkcije cilja, tj. optimalnog rješenja i algoritam će stati. Razvio ju je George Dantzig 1947. godine, a časopis *Computing in Science and Engineering* ju je uvrstio u top 10 algoritama 20.st.

U ovom poglavlju pretpostavljamo da imamo standardni oblik problema linearnog programiranja, da je $\mathcal{P} \subseteq \mathbb{R}^n$ dopustiv skup, $b \in \mathbb{R}^m$, $x, c \in \mathbb{R}^n$ te $A \in \mathbb{R}^{m\times n}$ sa linearno nezavisnim retcima.

Sljedeće tri definicije definiraju pojmove koji će se spominjati u ovom poglavlju.

Definicija 4.1. Za skup vektora $a_1, a_2, \ldots, a_n$ kažemo da je **linearno nezavisan** ako njihova proizvoljna linearna kombinacija iščezava jedino na trivijalan način, tj.

$$\lambda_1 a_1 + \cdots + \lambda_n a_n = 0 \Rightarrow \lambda_1 = \cdots = \lambda_n = 0.$$

Definicija 4.2. Za matricu A kažemo da je **regularna** ako postoji matrica B takva da vrijedi $AB = BA = I$.

Definicija 4.3. Rang matrice A je broj njenih linearno nezavisnih stupaca.

Definicija 4.4. Neka je $x \in \mathcal{P}$. Za vektor $d \in \mathbb{R}^n\backslash\{0\}$ kažemo da je **dopustiv smjer** u vektoru x ako postoji $\theta > 0$ takav da je $x + \theta d \in \mathcal{P}$.

Neka je $x = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ bazično dopustivo rješenje poliedra $\mathcal{P} = \{x : Ax = b, x \geq 0\}$. Neka je $B \in \mathbb{R}^{m\times m}$ matrica baze, te $B(1), \ldots, B(m)$ indeksi bazičnih varijabli. Bazični dio vektora x, x_B, dobijemo rješavanjem sustava $Bx_B = b$, a za nebazični dio vektora x vrijedi $x_i = 0$, $\forall i \in N := \{1, \ldots, n\}\backslash\{B(1), \ldots, B(m)\}$. Kako je x BDR, a u simpleks metodi "hodamo" po bridu poliedra koji predstavlja dopustivo područje od jednog bazičnog rješenja do drugog dok ne dostignemo optimalno rješenje, krenut ćemo od x u smjeru vektora d s duljinom koraka $\theta > 0$ i pritom moramo zadovoljiti uvjet optimalnosti ($c^T(x+\theta d) < c^T x$) te uvjet dopustivosti ($x + \theta d \in \mathcal{P}$). Imajući to na umu, možemo konstruirati vektor smjera. Odaberimo prvo jednu nebazičnu komponentu vektora x, npr. x_j, $j \in N$. Vektor d ćemo definirati na sljedeći način: $d_j = 1$, te $d_i = 0$, $\forall i \in N\backslash\{j\}$. Time smo definirali nebazični dio vektora d. Označimo bazični dio vektora d sa d_B i odredimo ga iz uvjeta dopustivosti: $Ax + \theta Ad = b \iff Ad = 0$.

$$Ad = \sum_{i=1}^{n} A_i d_i = \sum_{i=1}^{m} A_{B(i)} d_i + A_j = Bd_B + A_j = 0$$

Iz toga slijedi da je $d_B = -B^{-1}A_j$. Za ovako odabran d je zadovoljen uvjet $A(x + \theta d) = b$, sada još treba provjeriti je li zadovoljen uvjet nenegativnosti ($x + \theta d \geq 0$). Ako je x nedegenerativno bazično dopustivo rješenje, točno $n - m$ komponenti od x je jednako nuli. Za nebazični dio od $x + \theta d$ vrijedi $x_i + \theta d_i = 0$, $\forall i \in N\backslash\{j\}$ i $x_j + \theta d_j = 0$,

pa je zadovoljen uvjet nenegativnosti. Za bazični dio vektora x vrijedi $x_{B(i)}d_{B(i)} > 0$ ako je $d_{B(i)} > 0$. Ako je $d_{B(i)} < 0$, onda možemo odabrati dovoljno mali $\theta > 0$ tako da vrijedi $x_{B(i)}d_{B(i)} > 0$, pa je i u tom slučaju uvjet nenegativnosti zadovoljen, tj. $x + \theta d \in \mathcal{P}$. Ako je x degenerativno bazično dopustivo rješenje, više od $n - m$ komponenti je jednako nula. Za nebazični dio je zadovoljen uvjet nenegativnosti jednako kao u prvom slučaju. Za bazični dio vektora $x + \theta d$ postoji indeks i takav da je $x_{B(i)} = 0$, pa ako je $d_{B(i)} < 0$ onda ne postoji $\theta > 0$ takav da vrijedi $x_{B(i)} + \theta d_{B(i)} > 0$, pa tu nastaje problem, ali kasnije ćemo vidjeti kako izbjeći pojavljivanje tog slučaja.

Nadalje, trebamo vidjeti kako pravilno odabrati nebazičnu varijablu x_j. Tu će nam pomoći uvjet optimalnosti: $c^T(x+\theta d) < c^T x$, iz kojeg slijedi da c mora zadovoljavati $c^T d < 0$. Kako je $c^T d = c_B^T d_B + c_j = -c_B^T B^{-1} A_j + c_j$, slijedi da moramo odabrati j tako da bude zadovoljena nejednakost $c_j - c_B^T B^{-1} A_j < 0$. $c_j - c_B^T B^{-1} A_j$ ćemo označiti sa $\bar{c}_j$ i zvati *utjecaj nebazične varijable* x_j, a vektor $\bar{c} = [\bar{c_1}, \ldots, \bar{c_n}]^T \in \mathbb{R}^n$ vektor utjecaja. Utjecaji bazičnih varijabli su uvijek jednaki nuli.

Sljedeći teorem nam daje uvjete optimalnosti rješenja:

Teorem 4.1. *Neka je x BDR, a B odgovarajuća matrica baze i $\bar{c}$ vektor utjecaja.*

- *Ako je $\bar{c} \geq 0$, onda je x optimalno rješenje.*
- *Ako je x optimalno nedegenerativno rješenje, onda je $\bar{c} \geq 0$.*

Sada još samo preostaje odrediti θ, odnosno duljinu koraka u odabranom smjeru. Logično je da ako krenemo u nekom smjeru, idemo u tom smjeru dok god ne dođemo do nekog vrha poliedra, tj. $\theta^* = \max\{\theta : x + \theta d \geq 0\}$. Ako je $d \geq 0$ slijedi da je $x + \theta d \geq 0$, $\forall \theta > 0$, pa je $\theta^* = \infty$, tj. funkcija cilja je neomeđena funkcija. Ako postoji indeks i takav da je $d_i < 0$, trebamo odabrati $\theta > 0$ za koji vrijedi $x_i + \theta d_i \geq 0$, pa iz toga slijedi da je $\theta \leq -\frac{x_i}{d_i}$. Zbog toga je $\theta^* = \min\limits_{\{i=1,\ldots,m | d_{B(i)}<0\}} -\dfrac{x_{B(i)}}{d_{B(i)}}$.

Neka je l indeks u kojem se postiže navedeni minimum. Tada vrijedi $x_{B(l)} + \theta^* d_{B(l)} = 0$.

Teorem 4.2. *Stupci $A_{B(i)}$, $i \neq l$, $i = 1, \ldots, m$ i A_j su linearno nezavisni te je stoga matrica*

$$\bar{B} = [A_{B(1)}, \ldots, A_{B(l-1)}, A_j, A_{B(l+1)}, \ldots, A_{B(m)}]$$

regularna. Vektor $x + \theta^* d$ je bazično dopustivo rješenje.

Jedna iteracija simpleks metode:

1. Početi sa bazom koja se sastoji od bazičnih vektora $A_{B(1)}, \ldots, A_{B(m)}$ i odgovarajućim bazičnim dopustivim rješenjem x.
2. Izračunati utjecaje $\bar{c}_j = c_j - c_B^T B^{-1} A_j$ za sve nebazične indekse $j \in N$. Ako su svi nenegativni, trenutno bazično dopustivo rješenje je optimalno i algoritam staje. U suprotnom, odabrati neki indeks j za koji je $\bar{c}_j < 0$.
3. Izračunati $d = -B^{-1} A_j =: -u$. Ako su sve komponente vektora d nenegativne, θ^* je ∞, a optimalna vrijednost funkcije cilja je $-\infty$ i algoritam staje.

4. Ako je neka komponenta vektora d negativna, θ^* odredimo iz uvjeta

$$\theta^* = \min_{\{i=1,\dots,m|d_{B(i)}<0\}} -\frac{x_{B(i)}}{d_{B(i)}} = \min_{\{i=1,\dots,m|u_i>0\}} \frac{x_{B(i)}}{u_i}$$

5. Neka je l indeks za koji se postiže minimum iz koraka 4. Tada je $\theta^* = \frac{x_{B(l)}}{u_l}$. Formiramo novu bazu zamjenjujući stupac $A_{B(l)}$ sa stupcem A_j. Novo bazično rješenje $y = x + \theta^* d$ je dano s: $y_j = \theta^*$, $y_{B(i)} = x_{B(i)} + \theta^* d_{B(i)} = x_{B(i)} - \theta^* u_i$, te $y_k = 0$, $\forall k \in N \backslash \{j\}$.

4.1 Tableau implementacija simpleks metode

Uvjete LP problema možemo zapisati u obliku matrice $B^{-1}[b|A]$. Takva proširena matrica se naziva *simpleks tableau*. Stupac $B^{-1}b$, koji se naziva *nulti stupac*, sadrži vrijednosti bazičnih varijabli. Stupac $B^{-1}A_i$ zovemo i - ti stupac tableaua. Stupac $u = B^{-1}A_j$ koji odgovara varijabli koja ulazi u bazu se naziva *pivot stupac*. Ako l - ta bazična varijabla izlazi iz baze, tada se l - ti redak tableaua naziva *pivot redak*. Element na presjeku pivot retka i pivot stupca se naziva *pivot element*. Pivot element u_l je uvijek pozitivan (osim ako je $u \leq 0$, pa algoritam staje jer je postignut uvjet optimalnosti). Često se u simpleks tableau dodaje i *nulti redak* koji na prvom mjestu ima negativnu vrijednost funkcije cilja $-{c_B}^T x_B$, a ostali elementi su elementi vektora utjecaja $\bar{c}^T = c^T - c_B^T B^{-1} A$. Struktura simpleks tableaua je vidljiva na slici (2), odnosno na slici (3) po komponentama.

$-c_B^T B^{-1} b$	$c^T - c_B^T B^{-1} A$
$B^{-1}b$	$B^{-1}A$

Slika 2: Simpleks tableau

$-c_B^T x_B$	$\bar{c}_1$	$\dots$	$\bar{c}_n$
$x_{B(1)}$	|		|
$\vdots$	$B^{-1}A_1$	$\dots$	$B^{-1}A_n$
$x_{B(m)}$	|		|

Slika 3: Simpleks tableau po komponentama

Jedna iteracija simpleks tableau implementacije:

1. Napraviti tableau sa zadanom bazom B i odgovarajućim bazičnim dopustivim rješenjem x.

2. Pregledati utjecaje u nultom retku simpleks tableaua. Ako su svi nenegativni, tada je trenutno bazično dopustivo rješenje optimalno i algoritam staje. U suprotnom, odabrati neki indeks j za koji vrijedi $\bar{c}_j < 0$.

3. Pregledati vektor $u = B^{-1}A_j$, odnosno j - ti stupac simpleks tableaua. Ako niti jedna komponenta od u nije pozitivna, optimalna vrijednost funkcije cilja je $-\infty$ i algoritam staje.

4. Za svaki indeks i za koji je $u_i > 0$ izračunati omjer $x_{B(i)}/u_i$. Neka je l indeks za koji se postiže najmanji omjer. Stupac $A_{B(l)}$ izlazi iz baze, a stupac A_j ulazi u bazu.

5. Svakom retku tableaua dodati l - ti redak (pivot redak) pomnožen odgovarajućim skalarom kako bi pivot element postao 1, a svi ostali elementi u pivot stupcu 0.

U ovom postupku na više mjesta može doći do situacije da treba odabrati između dva ili više redaka koji su potencijalni kandidati za ulazak u bazu. Ako odaberemo "lošeg" kandidata, u postupku može doći do degenerativnog BDR-a ili do ciklusa. Kako do toga ne bi došlo Robert Bland je 1977. godine uveo pravilo za pivotiranje danas poznato pod nazivom Blandovo pravilo koje glasi:

1. Među svim nebazičnim varijablama koje su pogodne za ulazak u bazu odaberemo onu s najmanjim indeksom.

2. Među svim bazičnim varijablama koje su pogodne za izbacivanje iz baze odaberemo onu s najmanjim indeksom.

Sljedeći teorem potvrđuje da ćemo ovim pravilom za pivotiranje uvijek doći do rješenja:

Teorem 4.3. *Ako krenemo od nedegenerativnog bazičnog dopustivog rješenja i koristimo Blandovo pravilo, simpleks metoda će završiti u konačno mnogo koraka.*

4.2 Konstrukcija početnog BDR-a za simpleks metodu

U prvom koraku iteracije simpleks tableau metode stoji da treba odabrati bazu i odgovarajuće BDR. U ovom potpoglavlju ćemo vidjeti kako se najlakše odabere početno bazično dopustivo rješenje. Ako imamo LP problem zadan sa:

$$\begin{gathered} c^T x \to \min \\ Ax \leq b, \quad b \geq 0 \\ x \geq 0 \end{gathered}$$

Onda ga možemo zapisati na sljedeći način:

$$\begin{gathered} c^T x \to \min \\ Ax + s = b \\ x, s \geq 0 \end{gathered}$$

tj. u matričnom zapisu:

$$\begin{gathered} c^T x + 0^T s \to \min \\ \begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = b \\ \begin{bmatrix} x \\ s \end{bmatrix} \geq 0 \end{gathered}$$

Sada se jasno vidi da za matricu baze B možemo odabrati jediničnu matricu, pa iz $Bx_B = b$ slijedi da je $x_B = b \geq 0$ bazično dopustivo rješenje. No, postoje i slučajevi kada početno BDR nije tako očito. Pogledajmo sljedeći LP problem:

$$\left.\begin{array}{rcl} c^T x & \to & \min \\ Ax & = & b \\ x & \geq & 0 \end{array}\right\} \tag{8}$$

Bez smanjenja općenitosti možemo pretpostaviti da je $b \geq 0$. U svrhu određivanja početnog BDR-a uvodimo novi vektor $y = [y_1, \ldots, y_m]^T \in \mathbb{R}^m$, $y \geq 0$ koji se zove *vektor artificijelnih varijabli*. Sada možemo definirati novi LP problem:

$$\left.\begin{array}{rcl} y_1 + \cdots + y_m & \to & \min \\ Ax + y & = & b \\ x, y & \geq & 0 \end{array}\right\} \tag{9}$$

Za problem (9) početno bazično dopustivo rješenje je b. Ako je x bazično dopustivo rješenje problema (8), onda je $\begin{bmatrix} x & 0 \end{bmatrix}^T$ optimalno rješenje problema (9) i vrijednost funkcije cilja tog problema je 0. Ako je optimalna vrijednost funkcije cilja problema (9) različita od 0, tada problem (8) nema rješenja. Naposlijetku, ako je $y^* = 0$, tj. optimalna vrijednost funkcije cilja problema (9) jednaka nuli, onda je x^* BDR problema (8).
Pogledajmo sve do sada navedeno na jednom konkretnom primjeru:

Primjer 4.1. Zadan je LP problem:

$$\begin{gathered} -x_1 - 2x_2 \to \min \\ x_1 + x_2 \geq 5 \\ x_1 + x_2 \leq 8 \\ x_1 \leq 6 \\ x_2 \leq 6 \\ x_1, x_2 \geq 0 \end{gathered}$$

Prvo ga trebamo pretvoriti u standardni oblik LP problema:

$$-x_1 - 2x_2 \to \min$$
$$x_1 + x_2 - x_3 = 5$$
$$x_1 + x_2 + x_4 = 8$$
$$x_1 + x_5 = 6$$
$$x_2 + x_6 = 6$$
$$x_1, \ldots, x_6 \geq 0$$

Kako početni BDR nije očigledan, trebamo uvesti vektor artificijelnih varijabli, y, i definirati pomoćni problem:

$$y_1 + y_2 + y_3 + y_4 \to \min$$
$$x_1 + x_2 - x_3 + y_1 = 5$$
$$x_1 + x_2 + x_4 + y_2 = 8$$
$$x_1 + x_5 + y_3 = 6$$
$$x_2 + x_6 + y_4 = 6$$
$$x_1, \ldots, x_6, y_1, \ldots, y_4 \geq 0$$

U ovom problemu očigledno za matricu baze B možemo uzeti jediničnu matricu $I \in \mathbb{R}^{4\times 4}$, pa je početni BDR za pomoćni problem jednak $b = [5, 8, 8, 6]^T$. Sada ćemo uz pomoć simpleks tableaua pronaći rješenje pomoćnog LP problema:

-25	-3	-3	1	-1	-1	-1	0	0	0	0
$y_1 = 5$	1	1	-1	0	0	0	1	0	0	0
$y_2 = 8$	1	1	0	1	0	0	0	1	0	0
$y_3 = 6$	1	0	0	0	1	0	0	0	1	0
$y_4 = 6$	0	1	0	0	0	1	0	0	0	1

Zbog Blandovog pravila, 1. stupac je pivot stupac, a 1. redak pivot redak. To znači da y_1 izlazi iz baze, a x_1 ulazi u bazu. Nakon poništavanja elemenata u pivot stupcu pivot elementom dobijemo tablicu:

-10	0	0	-2	-1	-1	-1	3	0	0	0
$x_1 = 5$	1	1	-1	0	0	0	1	0	0	0
$y_2 = 3$	0	0	1	1	0	0	-1	1	0	0
$y_3 = 1$	0	-1	1	0	1	0	-1	0	1	0
$y_4 = 6$	0	0	0	0	0	1	0	0	0	1

Sada y_3 izlazi iz baze, a x_3 ulazi u bazu, pa nakon poništavanja elemenata u pivot stupcu pivot elementom dobivamo:

-8	0	-2	0	-1	1	-1	1	0	2	0
$x_1 = 6$	1	0	0	0	1	0	0	0	1	0
$y_2 = 2$	0	1	0	1	-1	0	0	1	-1	0
$x_3 = 1$	0	-1	1	0	1	0	-1	0	1	0
$y_4 = 6$	0	1	0	0	0	1	0	0	0	1

y_2 izlazi iz baze, a x_2 ulazi u bazu, pa nakon poništavanja elemenata u pivot stupcu pivot elementom dobivamo:

-4	0	0	0	1	-1	-1	1	2	0	0
$x_1 = 6$	1	0	0	0	1	0	0	0	1	0
$x_2 = 2$	0	1	0	1	-1	0	0	1	-1	0
$x_3 = 3$	0	0	1	1	0	0	-1	1	0	0
$y_4 = 4$	0	0	0	-1	1	1	0	-1	1	1

Sada y_4 izlazi iz baze, a x_5 ulazi u bazu, pa nakon poništavanja elemenata u pivot stupcu pivot elementom dobivamo:

0	0	0	0	0	0	0	1	1	1	1
$x_1 = 2$	1	0	0	1	0	-1	0	1	0	-1
$x_2 = 6$	0	1	0	0	0	1	0	0	0	1
$x_3 = 3$	0	0	1	1	0	0	-1	1	0	0
$x_5 = 4$	0	0	0	-1	1	1	0	-1	1	1

Nakon ovog koraka imamo $\bar{c} \geq 0$, pa algoritam staje i dobivamo optimalno rješenje s vrijednošću funkcije cilja 0, što znači da je $x^* = [2, 6, 3, 0, 4, 0]^T$ BDR za početni problem. Sada možemo zapisati tableau implementaciju početnog problema:

14	0	0	0	1	0	1
$x_1 = 2$	1	0	0	1	0	-1
$x_2 = 6$	0	1	0	0	0	1
$x_3 = 3$	0	0	1	1	0	0
$x_5 = 4$	0	0	0	-1	1	1

Kako je i ovdje $\bar{c} \geq 0$ dobili smo da je $x^* = [2, 6, 3, 0, 4, 0]^T$ optimalno rješenje početnog problema, a -14 minimalna vrijednost funkcije cilja.

4.3 Asimptotska vremenska složenost simpleks algoritma

Prilikom proučavanja algoritama uvijek je bitno vidjeti i koje je vrijeme izvršenja tog algoritma. Na taj način možemo vidjeti koliko je algoritam efikasan i usporediti njegovu brzinu i efikasnost sa drugim algoritmima koji možda rješavaju isti problem, te na taj način odlučiti koji je algoritam bolje koristiti. Pogledajmo prvo definiciju vremena izvršenja algoritma, te definiciju O notacije.

Definicija 4.5. Algoritam je konačan skup naredbi (aritmetičke operacije, usporedbe, uvjetne izjave, instrukcije za upis/ispis podataka itd.) koje se koriste u programskim jezicima. **Vrijeme izvršenja algoritma** je broj naredbi koje se izvršavaju u algoritmu.

Vrijeme izvršenja algoritma (VIA) je funkcija koja ovisi o veličini inputa problema.

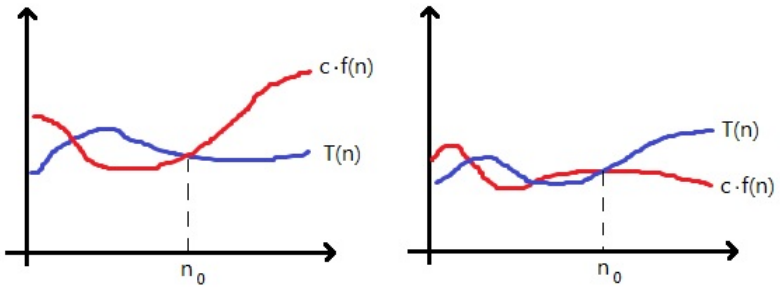
Definicija 4.6. Neka su $T, f : \mathbb{N} \to \mathbb{R}^+$ funkcije. $T(n) = \mathrm{O}(f(n))$ ako postoje pozitivne konstante $n_0 \in \mathbb{N}$ i $c \in \langle 0, \infty \rangle$ takve da vrijedi

$$T(n) \leq c \cdot f(n), \quad \forall n \geq n_0.$$

$T(n) = \Omega(f(n))$ ako postoje konstante $n_0 \in \mathbb{N}$ i $c \in \langle 0, \infty \rangle$ takve da vrijedi

$$T(n) \geq c \cdot f(n), \quad \forall n \geq n_0.$$

$T(n) = \Theta(f(n))$ ako je $T(n) = \mathrm{O}(f(n))$ i $T(n) = \Omega(f(n))$.



Slika 4: Lijevo: grafički prikaz O notacije. Desno: grafički prikaz Ω notacije.

Pogledajmo sada i definiciju polinomijalnog vremena izvršenja algoritma.

Definicija 4.7. Kažemo da algoritam ima **polinomijalno** vrijeme izvršenja ako napravi $\mathrm{O}\left(n^k\right)$ elementarnih operacija na racionalnim brojevima veličine $\mathrm{O}\left(n^k\right)$ za neki $k \in \mathbb{N}$, gdje je n duljina inputa problema.

Napomena. Veličina nekog broja x je broj bitova potrebnih za binarni zapis tog broja. Pretpostavimo sada da je $A \in \mathbb{Q}^{m \times n}$, $c \in \mathbb{Q}^n$ i $b \in \mathbb{Q}^n$.

Tada za jednu iteraciju simpleks metode (uz pretpostavku da je poznat inverz matrice baze) imamo $\mathrm{O}(m \cdot n)$ operacija, što je polinomijalno vrijeme izvršenja. Sljedeći teorem daje ogradu za vrijeme izvršenja jedne iteracije simpleks algoritma.

Teorem 4.4. *Jedna iteracija simpleks algoritma zahtjeva ukupno* $\mathrm{O}(m \cdot n)$ *operacija na racionalnim brojevima čija je veličina polinomijalna u veličini inputa.*

Sada se postavlja pitanje koliko iteracija ima u simpleks metodi? Broj vrhova poliedra koji odgovara području definiranom uvjetima LP problema može rasti eksponencijalno s brojem varijabli i uvjeta. Za mnoga pravila pivotiranja se ispostavilo da je donja granica na iteracije eksponencijalna (Klee i Minty, 1972.), no u praksi, simpleks metoda obično završi u $\mathrm{O}(m)$ iteracija. Prosječno vrijeme izvršenja simpleks metode (u određenim probabilističkim problemima) je polinomijalno (Bogwardt, 1982.). Ako svakom koeficijentu nekog proizvoljnog LP problema dodamo normalnu slučajnu varijablu sa očekivanjem 0 i varijancom σ^2, onda je očekivani broj iteracija simpleks metode polinomijalan u terminima $\frac{1}{\sigma}$, n i m (Spielman i Teng, 2004.). Trenutno otvoreni problem o kojem se raspravlja je postoji li pravilo pivotiranja za simpleks algoritam koje bi za svaki LP problem rezultiralo polinomijalnim brojem iteracija? Taj problem se čak nalazi na Smaleovoj listi: Matematički problemi sljedećeg stoljeća. (Smale, 1998.) Zasad su najbolje rezultate dali Kalai (1992., 1997.) i Matoušek,

Sharir i Welzl (1996.) koji su dali "slučajna"[2] pravila pivotiranja s očekivanim brojem od $2^{O(\sqrt{m})}$ iteracija, a Friedmann, Hansen i Zwick su 2011. godine dali gotovo odgovarajuće donje granice za ta pravila.

[2]Slučajnost se odnosi na činjenicu da matrica baze jedne iteracije koja slijedi iz prethodne iteracije, nije određena deterministički nego ima veze sa slučajnim varijablama.

5 Dualnost

U teoriji optimizacije, dualnost znači da se neki problem može promatrati s dva različita gledišta: kao primalni i kao dualni problem. Teorija dualnosti je jako dobar teoretski alat koji ima brojne primjene, daje nove geometrijske uvide u zadane probleme i dovodi do još jednog algoritma za rješavanje problema linearnog programiranja: primal - dual simpleks metode koju ćemo vidjeti na kraju ovog poglavlja. Teoriju dualnosti možemo promatrati kao neku vrstu proširenja Lagrangeove metode za traženje minimuma funkcije uz zadana ograničenja (odnosno uvjete). Za problem minimizacije funkcije $f(x) = x^2 + y^2$ uz uvjet $x + y = 1$ uvedemo Lagrangeov multiplikator p i dobijemo $L(x, y, p) = x^2 + y^2 + p(1 - x - y)$. Sada, uz fiksan p možemo minimizirati funkciju $L(x, y, p)$ po x i y. Optimalno rješenje ovog problema je $x = y = p/2$ i ovisi o p. Sada iskoristimo uvjet s početka $x + y = 1$ i dobijemo optimalno rješenje zadanog problema: $x = y = 1/2$. Glavna ideja ovog pristupa je da ne tražimo odmah u početku da uvjet $x + y = 1$ bude zadovoljen, nego uvodimo cijenu p uz taj uvjet koji može biti i narušen. Kad se p pravilno odabere (u ovom slučaju $p = 1$), optimalno rješenje polaznog problema je jednako optimalnom rješenju problema bez uvjeta, tj. uz pravilan izbor parametra p, postojanje ili odsustvo početnih uvjeta ne igra nikakvu ulogu u optimalnoj vrijednosti funkcije cilja. U problemima linearnog programiranja možemo primijeniti isti taj pristup. Svaki od danih uvjeta u nekom LP problemu pomnožimo sa jednom varijablom i dodamo funkciji cilja te pokušavamo pronaći uz koje vrijednosti dodanih varijabli postojanje ili odsustvo početnih uvjeta ne utječe na optimalnu vrijednost funkcije cilja. Ispostavlja se da se prave vrijednosti dodanih varijabli mogu dobiti rješavanjem novog LP problema koji se naziva **dual** originalnog problema. Pogledajmo kako se dobiva forma dualnog problema.

Promotrimo standardni oblik LP problema: $c^T x \to \min$ uz uvjete $Ax = b, x \geq 0$ koji ćemo zvati primalnim problemom i pretpostavimo da postoji x^* koji je optimalno rješenje tog problema. Uvedimo sada relaksirani problem u kojem umjesto uvjeta $Ax = b$ u funkciju cilja dodajemo dodatni trošak $p^T(b - Ax)$ gdje p označava vektor cijene dimenzije jednake kao vektor b. Relaksirani problem je sljedećeg oblika: $c^T x + p^T(b - Ax) \to \min$ uz uvjet $x \geq 0$. Označimo sa $g(p)$ optimalnu vrijednost funkcije cilja relaksiranog problema. Taj problem dozovoljava puno više izbora dopustivih rješenja nego primalni problem i očekujemo da je vrijednost $g(p)$ manja ili jednaka optimalnoj vrijednosti funkcije cilja primala. Kako vrijedi $g(p) = \min_{x \geq 0}[c^T x + p^T(b - Ax)] \leq c^T x^* + p^T(b - Ax^*) = c^T x^*$ vidimo da je za svaki vektor p, $g(p)$ donja granica za optimalnu vrijednost primala $c^T x^*$. Problem $g(p) \to \max$ bez dodatnih uvjeta, se može interpretirati kao pronalaženje najveće moguće donje granice za optimalno rješenje primalnog problema i poznat je pod nazivom dualni problem.

Koristeći definiciju od $g(p)$ imamo:

$$g(p) = \min_{x \geq 0}[c^T x + p^T(b - Ax)] = p^T b + \min_{x \geq 0}(c^T - p^T A)x].$$

Uočimo da je

$$\min_{x \geq 0}(c^T - p^T A)x = \begin{cases} 0, & \text{ako je } c^T - p^T A \geq 0 \\ -\infty, & \text{inače} \end{cases}$$

Kada maksimiziramo $g(p)$ zapravo nam trebaju samo one vrijednosti vektora p za koje je $g(p)$ različit od $-\infty$, pa možemo zaključiti da je dualni problem jednak problemu:

$$p^T b \to \max$$

uz uvjet

$$p^T A \leq c^T.$$

Pogledajmo sada pravila za prebacivanje primala u dual i jedan konkretan primjer prebacivanja primalnog LP problema u dualni LP problem.

Neka je dana matrica A i neka su njeni retci označeni sa a_i^T, a stupci sa A_j. Ako je zadan primalni problem sa strukturom kao što je prikazana na lijevoj strani, dualni problem ima strukturu kao što je prikazana na desnoj strani:

	$c^T x \to \min$			$p^T b \to \max$	
uz uvjete	$a_i^T x \geq b_i,$	$i \in M_1$	uz uvjete	$p_i \geq 0,$	$i \in M_1$
	$a_i^T x \leq b_i,$	$i \in M_2$		$p_i \leq 0,$	$i \in M_2$
	$a_i^T x = b_i,$	$i \in M_3$		p_i slobodna,	$i \in M_3$
	$x_j \geq b_i,$	$j \in N_1$		$p^T A_j \leq c_j,$	$j \in N_1$
	$x_j \leq b_i,$	$j \in N_2$		$p^T A_j \geq c_j,$	$j \in N_2$
	x_j slobodna,	$j \in N_3$		$p^T A_j = c_j,$	$j \in N_3$

Primjer 5.1. Neka je zadan sljedeći LP problem:

$$x_1 + 2x_2 + 3x_3 \to min$$

uz uvjete

$$\begin{aligned} x_1 + 2x_2 &\geq 10 \\ x_1 + x_2 + 2x_3 &\geq 20 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Potrebno je pronaći dual danog problema. Ako slijedimo pravila za prebacivanje primala u dual dana iznad, dobivamo:

$$10p_1 + 20p_2 \to max$$

uz uvjete

$$\begin{aligned} p_1 + p_2 &\leq 1 \\ 2p_1 + p_2 &\leq 2 \\ 2p_2 &\leq 3 \\ p_1, p_2 &\geq 0 \end{aligned}$$

Vidimo da smo na ovaj način dobili problem koji ima manje varijabli nego primalni problem, a takve probleme je u načelu lakše riješiti.

Svaki primalni LP problem možemo pretvoriti u dualni problem, pa ako dobiveni problem shvatimo kao novi primal i njega možemo pretvoriti u dual te na taj način dobiti početni primalni problem. To nam potvrđuje i sljedeći teorem:

Teorem 5.1. *Ako pretvorimo dual u ekvivalentni minimizacijski problem i tada formiramo dual dobivenog problema, dobivamo problem ekvivalentan početnom primalnom problemu.*

Skraćena verzija iskaza teorema 5.1 glasi: dual duala je primal.

Navedimo i jedan rezultat koji će nam biti od pomoći prilikom dokazivanja jake dualnosti u sljedećem potpoglavlju.

Teorem 5.2. *Pretpostavimo da imamo LP problem* Π_1 *i da smo ga transformirali u LP problem* Π_2 *nizom transformacija sljedećih tipova:*

- *Zamijeniti slobodnu varijablu razlikom dviju nenegativnih varijabli.*
- *Zamijeniti nejednakosti jednakostima ubacujući nenegativnu pomoćnu varijablu.*
- *Ako je neki redak matrice A u dopustivom LP problemu u standardnom obliku linearna kombinacija drugih redaka, eliminirati odgovarajuću nejednakost iz uvjeta.*

Tada su duali od Π_1 *i* Π_2 *ekvivalentni, tj. oba duala su dopustiva i imaju jednake optimalne vrijednosti funkcije cilja.*

5.1 Teoremi dualnosti i komplementarni uvjeti

Od prije znamo da je $g(p)$ donja granica za optimalnu vrijednost funkcije cilja primala ako je primal bio problem u standardnom obliku. Sljedeći teorem nam govori da je $g(p)$ donja granica optimalne vrijednosti funkcije cilja za sve LP probleme.

Teorem 5.3. (slaba dualnost)
Ako je x dopustivo rješenje primalnog problema i p dopustivo rješenje dualnog problema, tada vrijedi:

$$p^T b \leq c^T x.$$

Dokaz. Za proizvoljne vektore x i p definirajmo: $u_i = p_i(a_i^T x - b)$ i $v_j = (c_j - p^T A_j)x_j$. Pretpostavimo da je x dopustiv za primalni problem, a p dopustiv za dualni problem. Iz definicije dualnog problema znamo da predznaci od p_i i $a_i^T x - b_i$ moraju biti jednaki, te da predznaci od x_j i $c_j - p^T A_j$ također moraju biti jednaki. Dakle, $u_i \geq 0, \forall i$ te $v_j \geq 0, \forall j$. Uočimo da vrijedi $\sum_i u_i = p^T Ax - p^T b$ i $\sum_j v_j = c^T x - p^T b$. Zbrajanjem tih dviju jednakosti i korištenjem nenegativnosti od u_i i v_j dobivamo:

$$0 \leq \sum_i u_i + \sum_j v_j = c^T x - p^T b.$$

□

Navedeni teorem ima sljedeće dvije korisne posljedice:

Korolar 5.1.
- *Ako je optimalna vrijednost funkcije cilja primala $-\infty$, tada je dualni problem nedopustiv.*
- *Ako je optimalna vrijednost funkcije cilja duala ∞, tada je primalni problem nedopustiv.*

Korolar 5.2. *Neka je x dopustivo rješenje primala, a p dopustivo rješenje duala i pretpostavimo da vrijedi $p^T b = c^T x$. Tada je x optimalno rješenje primala, a p optimalno rješenje duala.*

Sljedeći teorem je najvažniji teorem teorije dualnosti:

Teorem 5.4. (jaka dualnost)
Ako LP problem ima optimalno rješenje, tada i dual tog LP problema ima optimalno rješenje i ta dva rješenja su jednaka.

Dokaz. Pretpostavimo da imamo LP problem u standardnom obliku:

$$c^T x \to \min$$

uz uvjete

$$Ax = b$$
$$x \geq 0$$

Pretpostavimo i da su retci matrice A linearno nezavisni te da postoji optimalno rješenje danog problema. Primjenom simpleks metode, koristeći neko od pravila za izbjegavanje ciklusa, u konačno mnogo koraka dolazimo do optimalnog rješenja x i pripadajuće matrice baze B. Označimo sa $x_B = B^{-1}b$ odgovarajući vektor bazičnih varijabli. Kada simpleks metoda završi, optimalna vrijednost funkcije cilja mora biti nenegativna, pa slijedi $c^T - c_B^T B^{-1} A \geq 0$, gdje je sa c_B označen vektor čije su komponente cijene bazičnih varijabli. Definirajmo sada vektor p kao $p^T = c_B^T B^{-1}$. Tada imamo $p^T A \leq c^T$ što pokazuje da je p dopustivo rješenje dualnog problema

$$p^T b \to \max$$

uz uvjete

$$p^T A \leq c^T$$

Dodatno vrijedi $p^T b = c^T B^{-1} b = c^T x_B = c^T x$. Prema korolaru 5.2 slijedi da je p optimalno rješenje duala i optimalna vrijednost funkcije cilja duala je jednaka optimalnoj vrijednosti funkcije cilja primala.

Ako imamo proizvoljan LP problem koji ima optimalno rješenje, prvo ga pretvorimo u ekvivalentan LP problem u standardnom obliku čiji su retci matrice A međusobno linearno nezavisni. Duali početnog problema i problema u standardnom obliku su ekvivalentni (iz teorema 5.2), a kako smo pokazali da su optimalne vrijednosti funkcije cilja primala i duala jednake kada imamo problem u standardnom obliku, slijedi da i početni problem i njegov dual imaju jednaku optimalnu vrijednost funkcije cilja. □

Sljedeći teorem (o komplementarnim uvjetima) nam daje važnu poveznicu između optimalnih rješenja primala i duala.

Teorem 5.5. (komplementarni uvjeti)
Neka su x i p dopustiva rješenja primalnog i dualnog problema, redom. Vektori x i p su optimalna rješenja primala i duala ako i samo ako vrijedi:

$$p_i(a_i^T - b_i) = 0, \quad \forall i$$
$$(c_j - p^T A_j)x_j = 0, \quad \forall j$$

5.2 Primal - dual simpleks metoda

U dokazu teorema jake dualnosti primijenili smo simpleks metodu na primalni problem u standardnom obliku i definirali vektor p kao $p^T = c_B^T B^{-1}$. Tada smo uočili da je primalni uvjet optimalnosti $c^T - c_B^T B^{-1} A \geq 0$ jednak dualnom uvjetu optimalnosti $p^T A \leq c^T$. Dakle, simpleks metodu definiranu u polglavlju 4 možemo shvatiti kao algoritam koji održava dopustivost primala i kreće se prema dopustivosti duala. Metoda s tim svojstvom se općenito naziva primalni algoritam. Alternativni pristup je početi od dopustivog rješenja duala i kretati se prema dopustivosti primala. Metoda toga tipa se naziva dualni algoritam. U ovom poglavlju ćemo vidjeti kako izgleda dualni simpleks algritam. Pretpostavimo da imamo standardni oblik problema linearnog programiranja. Pogledajmo jednu iteraciju dualnog simpleks algoritma koristeći tableau kao što je prikazano na slikama (2) i (3):

1. Napraviti tableau sa zadanom bazom B, gdje su sve komponente vektora utjecaja nenegativne.

2. Promotriti komponente vektora $B^{-1}b$ u nultom retku tableaua. Ako su sve komponente nenegativne, došli smo do optimalnog rješenja i algoritam staje. U suprotnom odabrati indeks l za koji je $x_{B(l)} < 0$.

3. Promotriti l - ti redak tableaua (pivot redak) koji ima elemente $x_{B(l)}, v_1, \ldots, v_n$. Ako su svi $v_i \geq 0$ onda je optimalna vrijednost funkcije cilja duala ∞ i algoritam staje.

4. Za svaki i za koji vrijedi $v_i < 0$ izračunati omjer $\bar{c}_i/|v_i|$ i odabrati indeks j za koji se postiže najmanji omjer. Stupac $A_{B(l)}$ izlazi iz baze, a stupac A_j ulazi u bazu.

5. Svakom retku tableaua dodati l - ti redak (pivot redak) pomnožen odgovarajućim skalarom tako da pivot element v_j postane 1, a ostali elementi u pivot stupcu postanu jednaki 0.

Kao i u primalnom simpleks algoritmu, i u dualnom algoritmu može doći do ciklusa. No, ako slijedimo odgovarajuće pravilo, ciklusi se mogu izbjeći. Definirajmo prvo leksikografsko uređenje:

Definicija 5.1. Za vektor $u \in \mathbb{R}^n$ kažemo da je **leksikografski pozitivan (negativan)** ako je $u \neq 0$ i prvi nenegativni element od u je pozitivan (negativan). Pišemo $u \overset{L}{>} 0$ ($u \overset{L}{<} 0$).

Za vektor $u \in \mathbb{R}^n$ kažemo da je **leksikografski veći (manji)** od vektora $v \in \mathbb{R}^n$ ako je $u \neq v$ i $u - v \overset{L}{>} 0$ ($u - v \overset{L}{<} 0$). Pišemo $u \overset{L}{>} v$ ($u \overset{L}{<} v$).

Leksikografsko pravilo pivotiranja za dualnu simpleks metodu:

1. Odabrati bilo koji indeks l za koji vrijedi $x_{B(l)} < 0$ i postaviti l - ti redak za pivot redak.

2. Odrediti indeks j stupca koji će ući u bazu na sljedeći način: za svaki stupac za koji vrijedi $v_i < 0$ podijeliti sve elemente sa $|v_i|$ i odabrati leksikografski najmanji stupac. Ako postoji nekoliko leksikografski najmanjih stupaca, odabrati onaj s najmanjim indeksom.

Dualnu simpleks metodu možemo koristiti kada nam je bazično dopustivo rješenje dualnog problema već dostupno. Pretpostavimo da već imamo optimalnu bazu za neki LP problem i da sada želimo riješiti isti problem, ali sa drukčijim vektorom b. Tada postojeća optimalna baza ne mora nužno biti dopustiva za primalni problem. No, s druge strane, promjenom vektora b se ne mijenja vektor utjecaja $\bar{c}$, pa još uvijek imamo isto dualno dopustivo rješenje. Dakle, umjesto da moramo krenuti od samog početka, možemo iskoristi bazu koja je optimalna za originalni problem i dualnim simpleks algoritmom doći do optimalnog rješenja.

6 Problem cjelobrojnog linearnog programiranja

Problemi cjelobrojnog linearnog programiranja su vrlo teški za riješiti. Trenutno ne postoji niti jedan generalni algoritam koji efikasno rješava sve cjelobrojne probleme linearnog programiranja.

Definicija 6.1. Neka su dane matrice A, B i vektori b,c,d. Problem oblika

$$c^T x + d^T y \to \min$$

uz uvjete

$$Ax + By = b$$
$$x, y \geq 0$$
$$x \in \mathbb{Z}$$

zovemo **mješoviti problem linearnog programiranja**, a ako izbacimo neprekidnu varijablu y, dobivamo **cjelobrojni problem linearnog programiranja**. Ako su dodatno komponente vektora x iz skupa $\{0, 1\}$ onda govorimo o **nula - jedan**, odnosno **binarnom** problemu cjelobrojnog linearnog programiranja.

Uobičajeno je pretpostaviti i da su elementi matrica A i B i vektora b, c, d također cijeli brojevi.

6.1 Tehnike modeliranja

Postoji puno više stvarnih problema koji se mogu formulirati kao cjelobrojni problemi linearnog programiranja nego kao obični LP problemi, pa je zbog toga veoma korisno znati formulirati neki problem kao cjelobrojni problem linearnog programiranja. No, ne postoji sistematičan način za formulaciju takvih problema, nego za svaki primjer treba pronaći odgovarajući model. Pogledajmo neke primjere formuliranja cjelobrojnih LP problema.

Primjer 6.1. ***Binarni izbor***
Binarna varijabla x može poslužiti za kodiranje izbora između dviju alternativi. Pogledajmo sljedeći problem proizvodnje.

Pretpostavimo da imamo 3 projekta od kojih svaki traje 3 godine. Zarade od projekata su 0.2, 0.3 i 0.5 redom. Za prvi projekt je prve godine potrebno 0.5 jedinica kapitala, druge godine 0.3 jedinice i treće godine 0.2 jedinice. Za drugi projekt su te vrijednosti 1 prve godine, 0.8 druge godine i 0.2 treće godine. Treći projekt zahtjeva 1.5 jedinica kapitala prve i druge godine te 0.3 jedinice kapitala treće godine. Iznos dostupnog kapitala je ograničen, pa tako prve godine imamo 3.1 jedinicu kapitala na raspolaganju, druge godine 2.5 jedinica kapitala i treće godine 0.4 jedinice kapitala. Potrebno je odlučiti koje ćemo projekte započeti kako bismo maksimizirali zaradu, a da pri tome pazimo na zadana ograničenja.

Ovdje uvodimo binarnu varijablu odluke x_j koja je jednaka 1 ako smo odlučili započeti j - ti projekt te 0 ako nećemo započinjati j - ti projekt. Sada problem možemo formulirati na sljedeći način:

$$0.2x_1 + 0.3x_2 + 0.5x_3 \to \max$$

uz uvjete

$$\begin{aligned} 0.5x_1 + x_2 + 1.5x_3 &\leq 3.1 \\ 0.3x_1 + 0.8x_2 + 1.5x_3 &\leq 2.5 \\ 0.2x_1 + 0.2x_2 + 0.3x_3 &\leq 0.4 \\ x_j \in \{0,1\}, \quad j &= 1,2,3 \end{aligned}$$

Primjer 6.2. ***Vezani uvjeti***
U diskretnim optimizacijskim problemima je vrlo česta pojava da su određeni uvjeti međusobno zavisni. Npr. pretpostavimo da odluka A može biti donešena samo ako je donešena odluka B. Kako bismo modelirali takvu situaciju možemo uvesti binarne varijable x, odnosno y koje su jednake 1 ako je odluka A, odnosno B, donešena ili nula u suprotnom. Zavisnost između ovih dviju varijabli se može modelirati kao $x \leq y$, pa ako je $y = 0$ onda i x nužno mora biti 0. Pogledajmo to na sljedećem problemu.

Projekt menadžer u jednoj tvrtki razmatra skup od 10 velikih projektnih investicija. Investicije se razlikuju po dugoročnom profitu i po kapitalu koji je potreban da bi se pokrenule. Označimo sa P_j profit za j - tu investiciju, a sa C_j kapital potreban za pokretanje j - te investicije. Ukupan iznos dostupnog kapitala za ulaganje je Q. Prilike za investiranje 3 i 4 su međusobno isključive, kao i prilike 5 i 6. Nadalje, niti investicija 5 niti investicija 6 ne mogu biti ostvarene ako se ne ostvare investicije 3 i 4. Broj investicija koje se moraju pokrenuti iz skupa $\{1, 2, 7, 8, 9, 10\}$ je najmanje 2, najviše 4. Cilj projekt menađera je odabrati onu kombinaciju investicija koja će maksimizirati ukupan dugoročni profit uz dane uvjete.

Uvest ćemo binarnu varijablu x_j koja je jednaka 1 ako je odabrana j - ta investicija te 0 ako j - ta investicija nije odabrana. Kako su investicije 3 i 4 te 5 i 6 međusobno isključive dobivamo uvjete: $x_3 + x_4 \leq 1$ i $x_5 + x_6 \leq 1$. Vezane uvjete (5 i 6 se ne mogu pokrenuti ako nisu pokrenute 3 ili 4) zapisujemo na sljedeći način: $x_5 \leq x_3 + x_4$ i $x_6 \leq x_3 + x_4$. Sada naš problem možemo formulirati kao:

$$\sum_{j=1}^{10} P_j x_j \to \max$$

uz uvjete

$$\sum_{j=1}^{10} C_j x_j \leq Q$$
$$x_3 + x_4 \leq 1$$
$$x_5 + x_6 \leq 1$$
$$x_5 \leq x_3 + x_4$$
$$x_6 \leq x_3 + x_4$$
$$x_1 + x_2 + x_7 + x_8 + x_9 + x_10 \geq 2$$
$$x_1 + x_2 + x_7 + x_8 + x_9 + x_10 \leq 4$$
$$x_j \in \{0, 1\}, \quad j = 1, \ldots, 10$$

7 Problem određivanja optimalnog rasporeda

U sklopu praktičnog dijela ovog rada, u suradnji s tvrtkom IN2[3], kolegica Mateja Đumić i ja smo implementirale problem određivanja optimalnog rasporeda automobila za odlazak djelatnika tvrtke na razna događanja poput kongresa, seminara, stručnih usavršavanja itd. Cilj ovog praktičnog zadatka je bio riješiti jedan stvarni problem korištenjem znanja o linearnom programiranju i pokazati da linearno programiranje uistinu ima primjenu u stvarnom svijetu.

7.1 Opis problema

Tvrtka IN2 jako drži do stalnog usavršavanja svojih djelatnika. Zbog toga ih često šalje na razna događanja, od kojih su najčešća radionice i konferecije. Ako se radi o događanjima izvan mjesta rada i ako na tim događanjima sudjeluje veći broj sudionika, pojavljuje se potreba za određivanjem automobila koji idu na događanje te rasporedom ljudi po automobilima s ciljem što manjih troškova prijevoza. Prilikom pravljenja optimalnog rasporeda potrebno je uzeti u obzir sljedeće stvari:

- pojedini sudionici imaju na raspolaganju službeni automobil na trajnom korištenju i optimalno je koristiti to prijevozno sredstvo
- firma ima i nekoliko službenih automobila koje u ovisnosti i poslovnim potrebama mogu rezervirati i koristiti svi zaposlenici firme
- moguće je da zaposlenici koriste svoje privatne automobile, ako nema raspoloživih službenih automobila
- moguće je da zaposlenici koriste i javni prijevoz
- svaki zaposlenik može imati planiran boravak na događaju u svom periodu (ovisi npr. o terminu predavanja koji zaposlenik ima ili želi slušati, o privatnim razlozima, poslovnim razlozima itd.)

Ulazni parametri za optimizaciju su:

- popis sudionika događaja i grada iz kojeg sudionik polazi
- popis zaposlenika koji imaju službeni automobil na trajno korištenje
- popis službenih automobila koje mogu rezervirati svi zaposlenici i termini raspoloživosti
- organizacijska pripadnost zaposlenika (preferira se da zaposlenici iz iste organizacijske jedinice ili cjeline putuju zajedno) [4]
- raspored planiranog odlaska i planirani datum povratka

[3] http://www.in2.hr

[4] Struktura je hijerarhijska. Osnovna podjela je na firme, sektore i odjele. Sektori se mogu sastojati iz odjela i preferira se da zaposlenici iz istog odjela ili odjela istog sektora ili sektora iste firme putuju zajedno.

Izlazni rezultat je minimalan broj automobila potrebnih za odlazak na određenu radionicu ili konferenciju koji zadovoljava sve navedene uvjete. Uz broj potrebnih automobila, kao rezultat trebamo dobiti i raspored odlazaka i dolazaka tih automobila te raspored ljudi po automobilima. Cilj je, kao što je već i navedeno, dobiti minimalne troškove odlaska na događanje. Od sada pa nadalje, pretpostavimo da je događanje na koje djelatnici idu neki kongres.

7.2 Modeliranje problema

Kako je zadani problem cjelobrojni problem linearnog programiranja, modeliranje samog problema nije bio baš lak posao. Poznato je da su cjelobrojni LP problemi veoma teško rješivi i ako se u model stavi previše varijabli, postoji opasnost da se neće moći doći do rješenja u prihvatljivom vremenu. U procesu modeliranja isprobali smo nekoliko različitih modela u kojima su bile uključene i varijable tipa x_{ij} koje su označavale da i - ta osoba ide j - tim automobilom na kongres, pa je to automatski povlačilo i varijable tipa $x_{ij}^{'}$ koje označavaju da se i - ta osoba j - tim automobilom vraća sa kongresa, varijable tipa y_j koje su označavale koristi li se j - ti automobil za odlazak na kongres, te varijable tipa v_{jk} koje su označavale ide li j - ti automobil u k - to vrijeme na kongres, koje su odmah povlačile i varijable tipa $v_{ij}^{'}$ koje su označavale vraća li se j - ti automobil u k - to vrijeme s kongresa. Ovim načinom razmišljanja se već na malim uzorcima nakupio prevelik broj varijabli, pa smo ipak na kraju došli do jednostavnijeg modela s manje varijabli. Taj model će nam dati samo broj i raspored automobila za odlazak i dolazak s kongresa, a raspored osoba po automobilima će biti riješen jednim dodatnim jednostavnim algoritmom. Kako bismo uopće mogli napraviti model, uvedene su sljedeće pretpostavke:

- ima m djelatnika firme koji će sudjelovati na kongresu
- kongres traje r dana
- za odlazak na kongres i povratak s kongresa sudionik bira dan te hoće li to biti prijepodne ili poslijepodne (na taj način diskretiziramo vrijeme odlazaka i dolazaka te promatramo vremenske jedinice iz skupa $\{1, 2, \ldots, 2r\}$)
- ima $n + 2r$ vozila na raspolaganju, od toga je l službenih automobila na trajnom korištenju, h službenih automobila koje je moguće rezervirati, $n - l - h$ osobnih automobila te $2r$ vozila koja predstavljaju javni prijevoz u određenoj vremenskoj jedinici
- svaki od n automobila ima određeni kapacitet, dok vozila koja predstavljaju javni prijevoz imaju bekonačan kapacitet
- optimalni raspored zaposlenika iz različitih firmi te različitih gradova predstavlja nezavisne potprobleme

Varijable koje smo koristili u modelu su:

- $v_{jk},\ \ j \in \{1, \ldots, n\},\ \ k \in \{1, \ldots, 2r\}$ - varijabla koja označava ide li j - ti automobil u k - to vrijeme na kongres, poprima vrijednosti iz skupa $\{0, 1\}$, 0 ukoliko ne ide, odnosno 1 ukoliko ide.

- $v_{jk}^{'},\ j \in \{1,\ldots,n\},\ k \in \{1,\ldots,2r\}$ - varijabla koja označava vraća li se j - ti automobil u k - to vrijeme s kongresa, poprima vrijednosti iz skupa $\{0,1\}$, 0 ukoliko se ne vraća, odnosno 1 ukoliko se vraća.

- $a_k,\ k \in \{1,\ldots,2r\}$ - varijabla koja označava koristi li se javni prijevoz u k - toj jedinici vremena, poprima vrijednosti iz skupa $\{0,1\}$, 0 ukoliko niti jedan zaposlenik u k - tom vremenu ne koristi javni prijevoz, odnosno 1 ukoliko netko od zaposlenika putuje javnim prijevozom u k - tom vremenu

Za input modela imamo sljedeće oznake:

- $c_j,\ j \in \{1,\ldots,n\}$ - niz poznatih vrijednosti, predstavljaju kapacitet j - tog automobila.

- $z_{ik},\ i \in \{1,\ldots,m\},\ k \in \{1,\ldots,2r\}$ - niz poznatih konstanti, koje iznose 0 ili 1, 1 ukoliko i - ta osoba ide u k - to vrijeme, odnosno 0 ukoliko ne ide.

- $z_{ik}^{\prime},\ i \in \{1,\ldots,m\},\ k \in \{1,\ldots,2r\}$ - niz poznatih konstanti, koje iznose 0 ili 1, 1 ukoliko se i - ta osoba vraća s kongresa u k - to vrijeme, odnosno 0 u suprotnom.

Formulacija problema:

$$\sum_{j=1}^{n}\sum_{k=1}^{2r} v_{jk} + \sum_{k=1}^{2r} 2 \cdot a_k \to \min \tag{10}$$

uz uvjete:

$$\sum_{k=1}^{2r} v_{jk} \leq 1,\ \ j \in \{1,\ldots,n\} \tag{11}$$

$$\sum_{k=1}^{2r} v_{jk} - \sum_{k=1}^{2r} v_{jk}^{'} = 0,\ \ j \in \{1,\ldots,n\} \tag{12}$$

$$v_{jk} - \sum_{t=k}^{2r} v_{jt}^{'} \leq 0,\ \ j \in \{1,\ldots,n\}, k \in \{1,\ldots,2r\} \tag{13}$$

$$\sum_{i=1}^{m} z_{ik} \leq \sum_{j=1}^{n} v_{jk} \cdot c_j + a_k \cdot m,\ \ k \in \{1,\ldots,2r\} \tag{14}$$

$$\sum_{i=1}^{m} z_{ik}^{'} \leq \sum_{j=1}^{n} v_{jk}^{'} \cdot c_j + a_k \cdot m,\ \ k \in \{1,\ldots,2r\} \tag{15}$$

Funkcija koju želimo minimizirati je suma svih automobila koji idu na kongres. Druga suma u funkciji cilja obuhvaća javni prijevoz. Težine za sav javni prijevoz su 2 jer se preferira korištenje automobila tvrtke umjesto javnog prijevoza, pa na ovaj način osiguravamo da se prvo uzmu u obzir automobili u vlasništvu firme, a tek onda javni prijevoz. Model ima restrikciju da pojedini automobil smije samo jednom otići na kongres za vrijeme njegovog trajanja, pa uvjet (11) osigurava da automobil samo jednom ode na kongres. Ukoliko je suma u uvjetu (11) jednaka 1, to znači da je automobil otišao na kongres, a ako je suma

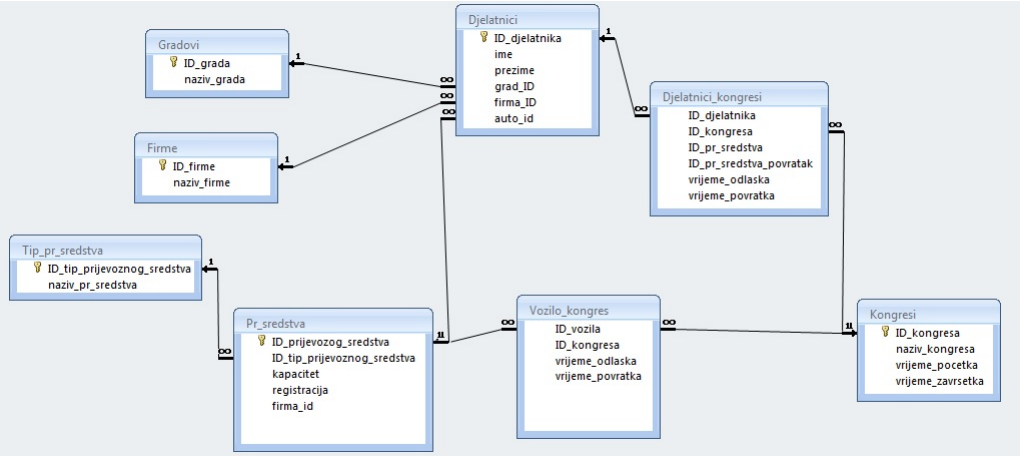
jednaka nula, to znači da automobil nije otišao na kongres. Uvjetom (12) osiguravamo da se automobil koji je otišao na kongres i vrati s kongresa. Uvjet (13) osigurava da se neki automobil ne može vratiti s kongresa ako na njega nije niti otišao. Npr. automobil ne može otići na kongres u 6. vremenskom periodu, a vratiti se u 2. Ovim uvjetom izbjegavamo upravo takve situacije. Uvjeti (14) i (15) osiguravaju da u svakom vremenskom trenutku koji promatramo broj osoba koje putuju na kongres ili sa kongresa ne prelaze kapacitet automobila koji putuju u promatranom vremenskom trenutku. Ako slučajno nema dovoljno slobodnih mjesta, onda i varijabla a_k postaje 1, što označava da se u k - tom trenutku koristi javni prijevoz s kapacitetom m koji ovdje predstavlja dovoljno velik broj (uočimo da ima m ljudi) da bi u javni prijevoz stali svi putnici koji imaju potrebu putovati javnim prijevozom.

7.3 Rješavanje problema

Kako bismo riješili postavljeni problem linearnog programiranja koristili smo program *Gurobi optimizer*. Gurobi je program specijaliziran za rješavanje problema linearnog programiranja. Uključuje programe za rješavanje sljedećih vrsta problema: problemi linearnog programiranja, problemi kvadratnog programiranja (eng. *quadratic programming problem*), problemi ograničenog kvadratnog programiranja (eng. *quadratically constrained programming problem*), problemi mješovitog linearnog programiranja, problemi cjelobrojno - kvadratnog programiranja (eng. *mixed-integer quadratic programming*) i problemi mješovitog cjelobrojno - kvadratnog ograničenog programiranja (eng. *mixed-integer quadratically constrained programming problem*). Metode koje Gurobi koristi prilikom rješavanja ovih problema su: primalni i dualni simpleks algoritmi, parallel barrier algorithm with crossover, concurrent optimization, shifting algorithm, deterministički algoritam, parallel branch-and-cut, non-traditional tree-of-trees search, višebrojne zadane heuristike (eng. *multiple default heuristics*), poboljšanje rješenja, cutting planes i detekcija simetrije. Kako Gurobi ima objektno - orijentirano sučelje prema raznim programskim jezicima, to smo i iskoristili te u programskom jeziku *C++* implementirali naš problem. Nakon unosa ulaznih parametara, Gurobi vraća točno vrijeme odlazaka i dolazaka određenih automobila na kongres. Kako bismo u potpunosti riješili zadani problem, uz pomoć dobivenog rasporeda i informacija o vremenima odlazaka ljudi na kongres i povratka s kongresa, "napunimo" automobile na odgovarajući način i dobijemo traženi optimalan raspored.

Podatke koji su potrebni kao input modela smo preko grafičkog sučelja napravljene aplikacije unosili u bazu podataka. Na slici (5) možemo vidjeti arhitekturu korištene baze podataka.

Baza se sastoji od osam tablica: Gradovi, Firme, Tip_pr_sredstva, Pr_sredstva, Djelatnici, Kongres te Djelatnici_kongres i Vozilo_kongres. U tablici *Gradovi* se nalaze podaci od imenima gradova, te o ID-u pojedinog grada. Isto tako, tablica *Firme* sadrži ID pojedinog unosa i njegov naziv. Tablica *Tip_pr_sredstva* sadrži ID tipa prijevoznog sredstva i naziv tog tipa. Tablica *Pr_sredstva* sadrži ID prijevoznog sredstva, ID tipa tog prijevoznog sredstva, registraciju, kapacitet, te ID firme u kojoj se vozilo nalazi. U tablici *Djelatnici* su sadržani svi podaci o djelatnicima: ID, ime, prezime, grad i firma kojoj djelatnik pripada te ID prijevoznog sredstva. ID prijevoznog sredstva je NULL ukoliko djelatniku nije dodijeljen niti jedan automobil na trajno korištenje, a ukoliko djelatnik ima osobni automobil ili službeni



Slika 5: Arhitektura baze podataka

automobil na trajnom korištenju, ovaj ID poprima vrijednost nekog od prijevoznih sredstava iz tablice *Pr_sredstva*. Isto tako, tablica *Kongresi* sadrži sve podatke o kongresima: ID, naziv, te dan početka i završetka kongresa. Na kraju, tu su tablice *Djelatnici_kongres* te *Vozilo_kongres* koje povezuje pojedinog djelatnika, odnosno vozilo s kongresom na kojem će prisustvovati. Tablica *Djelatnici_kongres* sadrži sljedeća polja: ID_djelatnika, ID_kongresa, ID prijevoznog sredstva koje će biti dodijeljeno djelatniku za odlazak na kongres, ID prijevoznog sredstva koje će biti dodijeljeno djelatniku za povratak sa kongresa, dan odlaska na kongres te dan povratka sa kongresa, dok tablica *Vozilo_kongres* sadrži polja: ID_vozila, ID_kongresa, vrijeme odlaska na kongres te vrijeme povratka sa kongresa.

Sama aplikacija je rađena koristeći *PHP*, *JavaScript*, *HTML* te *Gurobi* i *C++* za dobivanje optimalnog rasporeda. U aplikaciji postoje grafička sučelja za unos grada, firme i tipova prijevoznih sredstava u bazu. Dovoljno je samo unijeti naziv pojma i kliknuti gumb *Unesi*. Isto tako, postoji grafičko sučelje za unos prijevoznog sredstva u bazu koje sadrži padajuće izbornike za odabir tipa prijevoznog sredstva i firme, te polja za unos registracije i kapaciteta automobila. Unešeni podaci se klikom na gumb *Unesi* unose u odgovarajuću tablicu u bazi. Nadalje, u formi za unos djelatnika se unose ime i prezime djelatnika te se iz padajućih izbornika bira mjesto, firma te automobil koji je djelatniku dodijeljen, ako takav postoji. Ponovno se klikom na gumb unesi podaci spremaju u bazu. Nakon toga, postoji sučelje za dodjelu automobila. Prvo je potrebno iz padajućeg izbornika odabrati firmu, zatim djelatnika i na kraju automobil koji se dodjeljuje tom djelatniku. Ako djelatnik već ima dodijeljen automobil, aplikacija ga pita je li siguran da želi pregaziti taj izbor. Nakon potvrde, izmjene se spremaju u bazu. Forma za unos kongresa sadrži polje za unos naziva kongresa, te polja u koja se unose datumi i vremena početka i završetka kongresa. Forme djelatnici - kongres i vozilo - kongres sadrže izbornike u kojima se kongresu pridjeljuju djelatnici, odnosno vozila koji idu na kongres. Na kraju, ako u meniju odaberemo Upit, te odgovarajuću firmu i kongres, pozove se Gurobi koji izračuna i vrati optimalan raspored. Na slici (6) možemo vidjeti prikaz konzole u kojoj su ispisani podaci nakon rješavanja optimizacijskog problema. Gurobi je veoma brzo pronašao optimalno rješenje i kao što možemo vidjeti, pri tom je koristio simpleks metodu.

```
Optimize a model with 70 rows, 90 columns and 482 nonzeros
Found heuristic solution: objective 10
Presolve removed 64 rows and 78 columns
Presolve time: 0.00s
Presolved: 6 rows, 12 columns, 20 nonzeros
Variable types: 0 continuous, 12 integer (12 binary)

Root relaxation: objective 8.000000e+00, 7 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0       8.0000000    8.00000  0.00%     -    0s

Explored 0 nodes (7 simplex iterations) in 0.01 seconds
Thread count was 4 (of 4 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 8.000000000000e+00, best bound 8.000000000000e+00, gap 0.0%
```

Slika 6: Gurobi rezultat

Nakon toga se pojave dva gumba kojima možemo pogledati raspored za odlazak na kongres i raspored za povratak sa kongresa. Prikaz jednog rasporeda možemo vidjeti na slici (7).

Slika 7: Raspored djelatnika po automobilima

Djelatnici se mogu prevlačiti iz jednog automobila u drugi ako ti automobili putuju u istom vremenskom periodu te se sve osobe mogu prevući u javni prijevoz. Osobe označene crvenom bojom se ne mogu prevlačiti iz automobila u kojem jesu jer im je taj automobil dodijeljen i one su za njega odgovorne. Nakon eventualnih izmjena, klikom na gumb spremi se trenutno stanje sprema u bazu.

8 Literatura

[1] J.E.Beasley, OR-Notes. Dostupno na:
`http://people.brunel.ac.uk/~mastjjb/jeb/or/moreip.html`

[2] D. Bertsimas, J. N. Tsitsiklis, *Introduction to Linear Optimization*, Massachusetts Institute of Tehnology, Massachusetts, **2008.**

[3] F. Eisenbrand, *Online Coursera course - Linear and Discrete Optimisation*, École Polytechnique Fédérale de Lausanne, **2013.**

[4] Linear programming, Wikipedia. Dostupno na:
`http://en.wikipedia.org/wiki/Linear_programming`

[5] Simplex algorithm, Wikipedia. Dostupno na:
`http://en.wikipedia.org/wiki/Simplex_algorithm`

[6] Simplex method, Wolfram Math World. Dostupno na:
`http://mathworld.wolfram.com/SimplexMethod.html`

9 Sažetak

U svakodnevnom životu, ljudi često pokušavaju na neki način maksimizirati svoje dobitke i minimizirati gubitke. Zbog toga nije niti čudno da se linearno programiranje od svojih početaka u drugom svjetskom ratu do danas munjevito razvijalo. Problemi linearnog programiranja nam pomažu modelirati svakodnevne probleme i pronaći optimalna rješenja ovisno o ograničenjima koja su zadana. Do danas su razvijeni mnogi algoritmi koji se bave rješavanjem ovakvih problema. Jedan od njih, simpleks algoritam, je čak uvršten u top 10 algoritama 20. stoljeća. U ovom radu obrađujemo sam problem linearnog programiranja, geometriju linearnog programiranja, simpleks metodu, teoriju dualnosti te se dotičemo problema cjelobrojnog linearnog programiranja. Linearno programiranje uistinu ima primjenu u stvarnom životu, što dokazuje i konkretan problem optimalnog rasporeda vozila za putovanja na razna događanja koji je modeliran kao problem cjelobrojnog linearnog programiranja u suradnji sa kolegicom Matejom Đumić i gospodinom Ninoslavom Čerkezom iz tvrtke IN2, d.o.o. i detaljnije raspisan u posljednjem poglavlju rada.

10 Title and summary

Linear programming and applications. Most people try to somehow maximize their own gains and minimize losses in everyday life. Because of that fact, the linear programming has been developing very fast since it's beginnings in WW2 and it's still developing today. Linear programming problems can help us to model everyday problems and find optimal solutions for that problems depending on given constraints. To this day many algorithms have been developed to help us solve that kind of problems. The simplex algorithm was even included in top 10 algorithms of the 20^{th} century. In this paper we deal with following areas: linear programming problem, geometry of linear programming problem, simplex algorithm, duality theory and integer programming problem. In the end, we show that linear programming really can be applied in real - life problems by modelling the problem of optimizing the necessary number of cars for going to some event in cooperation with a colleague Mateja Đumić and Mr. Ninoslav Čerkez from IN2 company.

11 Životopis

Rođena sam 28. prosinca 1989. u Našicama. Osnovnu školu sam završila u Đuđenovcu te upisala prirodoslovno matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja sam sudjelovala na školskim i županijskim natjecanjima iz matematike, biologije i kemije. Po završetku srednje škole, 2008. sam upisala Sveučilišni preddiplomski studij matematike na Odjelu za matematiku u Osijeku. Preddiplomski studij sam uspješno završila i 2011. godine upisala Sveučilišni diplomski studij matematike, smjer Financijska i poslovna matematika.

Od listopada 2012. godine do veljače 2013. godine sam radila kao demonstrator na Ekonomskom fakultetu u Osijeku. U suradnji s kolegama s fakulteta sam držala razne radionice na festivalima znanosti od 2011. do 2013. Također, sudjelovala sam na Zimskoj školi matematike 2011. godine te u pripremanju učenika srednjih škola za natjecanja iz matematike (2013.). Od kolovoza do listopada 2013. godine sam bila na studentskoj praksi u Siemens CVC u Osijeku.