Metoda sjekućih ravni. Gomorijev rez.

Ema Djedović

Odsjek za matematičke i kompjuterske nauke
Prirodno-matematički fakultet
Univerzitet u Sarajevu

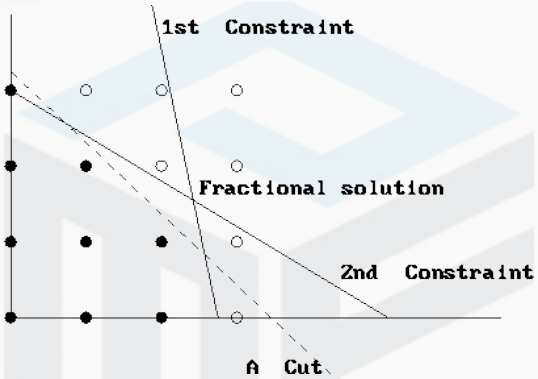
06/2024

Uvod

Algoritam sjekućih ravni rješava cjelobrojne programe modificirajući rješenja linearnih programa dok se ne dobije cjelobrojno rješenje. Ne dijeli dopušteno područje na podpodručja, kao u pristupima grananja i ograničavanja, već radi s jednim linearnim programom koji se rafinira **dodavanjem novih ograničenja**.

Nova ograničenja sukcesivno smanjuju dopušteno područje dok se ne pronađe optimalno cjelobrojno rješenje. Iako su u praksi postupci grananja i ograničavanja gotovo uvijek efikasniji, algoritam sjekućih ravni je bio važan za evoluciju cjelobrojnog programiranja. Historijski gledano, to je bio prvi algoritam za koji se moglo dokazati da **konvergira u konačno mnogo koraka** i koji je doveo do drugih, efikasnijih algoritama.

Uvod



Slika: Linearni program s realnim rješenjem (**Fractional solution**) na sjecištu prvog i drugog ograničenja. Cilj je doći do cjelobrojnih rješenja istaknutih crnim tačkama. U tu svrhu dodajemo jedan po jedan rez kako bismo "odsjekli" realna rješenja.

Primjer

Maksimizirati $Z = 7x_1 + 9x_2$

$$\begin{aligned} -x_1 + 3x_2 &\leq 6 \\ 7x_1 + x_2 &\leq 35 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

$\longrightarrow$

Maksimizirati $Z = 7x_1 + 9x_2$

$$\begin{aligned} -x_1 + 3x_2 + s_1 &= 6 \\ 7x_1 + x_2 + s_2 &= 35 \\ x_1, x_2, s_1, s_2 &\geq 0 \\ x_1, x_2, s_1, s_2 &\in \mathbb{Z} \end{aligned}$$

Bit će nam od koristi da izrazimo s_1 i s_2 preko varijabli x_1 i x_2:

$$\begin{aligned} s_1 &= 6 + x_1 - 3x_2 \\ s_2 &= 35 - 7x_1 - x_2 \end{aligned}$$

Primjer

Ako zanemarimo cjelobrojnost, dobit ćemo sljedeću optimalnu simplex tablicu:

Variable	x_1	x_2	s_1	s_2	$-z$	RHS
x_2	0	1	7/22	1/22	0	7/2
x_1	1	0	-1/22	3/22	0	9/2
$-z$	0	0	28/11	15/11	1	63

Posmatrajmo **prvo ograničenje**:

$$x_2 = \frac{7}{22}s_1 + \frac{1}{22}s_2 = \frac{7}{2}$$

Stavimo sve cijele dijelove na lijevu stranu, a sve razlomke na desnu:

$$x_2 - 3 = \frac{1}{2} - \frac{7}{22}s_1 - \frac{1}{22}s_2$$

Primjer

$$x_2 - 3 = \frac{1}{2} - \frac{7}{22}s_1 - \frac{1}{22}s_2$$

Sada, budući da lijeva strana sadrži samo cijele brojeve, desna strana također mora biti cjelobrojna. Kako imamo neki pozitivni razlomak (u našem slučaju $\frac{1}{2}$) umanjen za niz nekih pozitivnih vrijednosti, tako desna strana može biti samo 0, -1, -2, ... Dakle, dobijamo sljedeće ograničenje:

$$\frac{1}{2} - \frac{7}{22}s_1 - \frac{1}{22}s_2 \leq 0$$

Uvrstimo vrijednosti $s_1 = 6 + x_1 - 3x_2$ i $s_2 = 35 - 7x_1 - x_2$.

Primjer

$$\frac{1}{2} - \frac{7}{22}(6 + x_1 - 3x_2) - \frac{1}{22}(35 - 7x_1 - x_2) \leq 0$$

Sređivanjem dobivamo $-3 + x_2 \leq 0$ odnosno $x_2 \leq 3$.

Ograničenje koje smo dobili nazivamo **rez**, dodajemo ga u naš linearni program i isti ponovo rješavamo. Postupak ponavljamo sve dok optimalne vrijednosti za varijable odluke ne postanu cijeli brojevi - tada algoritam staje.

Primjer

Rez smo mogli generirati i iz **drugog ograničenja**. Osmotrimo ponovo tablicu s početka:

Variable	x_1	x_2	s_1	s_2	$-z$	RHS
x_2	0	1	7/22	1/22	0	7/2
x_1	1	0	-1/22	3/22	0	9/2
$-z$	0	0	28/11	15/11	1	63

Ovdje moramo biti oprezni da dobijemo pravilne znakove.

$$x_1 - \frac{1}{22}s_1 + \frac{3}{22}s_2 = \frac{9}{2}$$

$$x_1 + \left(-1 + \frac{21}{22}\right)s_1 + \frac{3}{22}s_2 = 4 + \frac{1}{2}$$

$$x_1 - s_1 - 4 = \frac{1}{2} - \frac{21}{22}s_1 - \frac{3}{22}s_2$$

Primjer

$$x_1 - s_1 - 4 = \frac{1}{2} - \frac{21}{22}s_1 - \frac{3}{22}s_2$$

Odavdje dobivamo ograničenje (rez):

$$\frac{1}{2} - \frac{21}{22}s_1 - \frac{3}{22}s_2 \leq 0$$

Ponovo izrazimo s_1 i s_2 preko varijabli odluke:

$$\frac{1}{2} - \frac{21}{22}(6 + x_1 - 3x_2) - \frac{3}{22}(35 - 7x_1 - x_2) \leq 0$$

čijim se sređivanjem dobije:

$$x_2 \leq \frac{10}{3} \quad \text{odnosno} \quad x_2 \leq 3 \quad \text{(radi cjelobrojnosti)}$$

Generalizirano

Ako imamo ograničenje $x_k + \sum a_i x_i = b$, gdje b **nije cijeli broj**, možemo pisati $a_i = \lfloor a_i \rfloor + a'_i$, za neko $0 \leq a'_i < 1$, i $b = \lfloor b \rfloor + b'$ za neko $0 < b' < 1$:

$$x_k + \sum (\lfloor a_i \rfloor + a'_i) x_i = \lfloor b \rfloor + b'$$

$$x_k + \sum \lfloor a_i \rfloor x_i + \sum a'_i x_i = \lfloor b \rfloor + b'$$

$$x_k + \sum \lfloor a_i \rfloor x_i = \lfloor b \rfloor + b' - \sum a'_i x_i$$

$$x_k + \sum \lfloor a_i \rfloor x_i - \lfloor b \rfloor = b' - \sum a'_i x_i$$

Tako dobijamo rez:

$$b' - \sum a'_i x_i \leq 0$$

Ovo novo ograničenje dodajemo u linearni program i ponovo rješavamo problem.

Komentar

Metoda sjekućih ravni može garantirati pronalaženje optimalnog cjelobrojnog rješenja. Međutim, postoje neki nedostaci:

- Greška zaokruživanja može uzrokovati velike poteškoće: Je li to 3.000000001 stvarno 3, ili trebamo generirati rez?
- Kao što metoda grananja i ograničenje može generirati veliki broj podproblema, ova tehnika može generirati veliki broj ograničenja (rezova).

Primjer uz Python biblioteku `lippy`

```
import lippy as lp
# lippy je Python biblioteka za rjesavanje problema
# linearnog programiranja

c_vektor = [3, 3, 7]
a_matrica =
[[1, 1, 1],
 [1, 4, 0],
 [0, 0.5, 3]]
b_vektor = [3, 5, 7]

gomory = lp.CuttingPlaneMethod(c_vektor, a_matrica, b_vektor)
print("Rjesenje:", gomory.solve())
```

Izvori

- https://pypi.org/project/lippy
- https://mat.tepper.cmu.edu
- Marija Ivanović (2009) *Vežbe iz Operacionih istraživanja* Univerzitet u Beogradu, Matematički fakultet
- Bradley, S.P., Hax, A.C. and Magnanti, T.L. (1977) *Applied mathematical programming* Reading, Mass: Addison-Wesley.