Univerza v Ljubljani

Fakulteta za matematiko in fiziko

Finančni praktikum

Skoraj konstantne vsote trikotnih funkcij na zaprtem intervalu

Poročilo

Avtorja:
Neža Kržan
Oskar Vavtar

Mentorja:
prof. dr. Sergio Cabello
doc. dr. Janoš Vidali

Ljubljana, 5. december 2021

Kazalo

1 Uvod

V poročilu bova predstavila problem izračuna skoraj konstantne vsote trikotnih funkcij na zaprtem intervalu in kodo, s katero sva problem rešila. V začetku poročila sledi podroben opis problema ter ugotovitve glede njegovega reševanja. V nadaljevanu sledijo komentrji na kodo, predstavitev rezultatov in generiranje programa.

2 Navodilo

Imamo nabor n trikotnih funkcij. Izmed njih želimo izbrati podmnožico največ sedmih, da bo njihova vsota čim bolj konstantna na intervalu $[0, 1]$ - s tem mislimo, da je razlika med maksimalno in minimalno vrednostjo vsote najmanjša možna. Ali je metoda celoštevilskega linearnega programiranja primerna za ta problem? Lahko uporabimo kakšne druge metode? Kako bi sami generirali smiselne vhodne podatke za ta problem? Bi lahko na enak način obravnavali tudi druge funkcije, npr. preproste stopničaste funkcije?

2.1 Razmislek in potek dela

Očitno je, da je več aspektov tega problema mogoče posplošiti. Trikotne funkcije lahko nadomestimo s katerimi drugimi preprostimi funkcijami (npr. stopničastimi, pravokotnimi). Namesto 7 funkcij, jih lahko izberemo r, $r < n$. Namesto intervala $[0, 1]$ pa lahko izberemo poljuben zaprt interval $[a, b]$, $-\infty < a \leq b < \infty$.

Problem sva se odločila rešiti za dva tipa trikotnih funkcij:

- simetrične,
- splošne.

Problem sva nameravala reševati z metodo celoštevilskega linearnega programiranja, ki se je kasneje izkazala za neustrezno. Namesto tega sva uporabila metodo mešano-celoštevilskega programiranja, ki v dovoli tudi realnoštevilske omejitve.

Kot opisano v navodilu je "čim bolj konstantno" mišljeno kot čim manjša razlika med maksimalno in minimalno vrednostjo vsote funkcij na intervalu. Iz nabora funkcij $\{f_1, \ldots, f_n\}$ želimo torej izbrat tako podmnožico $\{g_1, \ldots, g_r\}$, $r \leq n$, da bo dosežen

$$\min_{\{g_1,\ldots,g_r\}\subseteq\{f_1,\ldots,f_n\}} \left(\max_{x\in[a,b]} \sum_{i=1}^{r} g_i - \min_{x\in[a,b]} \sum_{i=1}^{r} g_i \right)$$

na želenem zaprtem intervalu $[a, b]$ (v navodilih $[a, b] = [0, 1]$, $r = 7$). Očitno je, da zgoraj zapisano ne formulira linearnega programa – funkciji max in max nista linearni. Formulacijo je bilo zato potrebno še malo spremeniti. Pri tem sva si pomagala z dejstvom, da so trikotne funkcije, s katerimi imava opravka, odsekoma linearne – ekstreme lahko dosežejo le v točkah, kjer se prelomijo. Ta lastnost se zaradi linearnosti prenese tudi na njihovo vsoto. Ta lahko ekstreme doseže le v točkah, kjer se prelomi neka funkcija, ki je del te vsote.

2.2 Programsko okolje in implementacija

Za implementacijo problema sva se odločila za uporabo programskega jezika `Sage`, ker ima že vgrajeno podporo za celoštevilsko linearno programiranje. V osnovi sloni na programskem jeziku `Python`, z dodatno podporo za matematiko, nastal pa je kot alternativa programskemu jeziku `Mathematica`. Programirala sva na platformi `CoCalc`, kjer sva datoteke pretvorila v `Jupyter Notebooks` obliko. Za prikaz rezultatov pri eksperimentiranju sva uporabila `R`.

3 Ideja rešitve z linearnim programiranjem LP

Kot omenjeno v uvodu, je ideja naslednja: iz nabora funkcij $\{f_1, \ldots, f_r\}$ želimo torej izbrati tako podmnožico $\{g_1, \ldots, g_r\}$, $r \leq 7 < n$, da bo dosežen

$$\min_{\{g_1,\ldots,g_7\}\subseteq\{f_1,\ldots,f_n\}} \left(\max_{x\in[a,b]} \sum_{i=1}^{r} g_i - \min_{x\in[a,b]} \sum_{i=1}^{r} g_i \right)$$

na želenem zaprtem intervalu $[a, b]$. Glede na to, da imamo opravka s trikotnimi funkcijami, ki so odsekoma linearne, pa lahko problem poenostavimo. Iz odsekoma linearnosti tako funkcij kot njihove vsote sledi, da bo vsota lahko dosegla ekstrem le na robovih intervala ali na mestu prelomu ene izmed funkcij. Vsoto lahko zato namesto na celotnem intervalu $[a, b]$ ocenimo le na točkah preloma. Definiramo torej množico testnih točk $\mathcal{B} = \{x_1, \ldots, x_k\}$, v kateri so vsebovane točke preloma obravnavanih funkcij iz intervala $[a, b]$ ter robni točki $\{a\}$ in $\{b\}$. Da lahko v teh točkah ocenimo vsoto, moramo izračunati vrednosti vseh funkcij f_j, $j \in [n]$, v vseh točkah množice $\mathcal{B}$. Zgornjo formulacijo problema lahko prepišemo kot

$$\min \left(\max_{x_i\in\mathcal{B}} \sum_{j=1}^{n} f_j(x_i)v_j - \min_{x_i\in\mathcal{B}} \sum_{j=1}^{n} f_j(x_i)v_j \right),$$

kjer v_j definiramo kot

$$v_j \;=\; \begin{cases} 1\,; & f_j \in \{g_1, \ldots, g_r\}, \\ 0\,; & f_j \notin \{g_1, \ldots, g_r\}. \end{cases}$$

Zdaj se moramo znebiti še funkcij max in min. Izračunamo lahko vrednost $\sum_{j=1}^{n} f_j(x_i)v_j$ za vsak i in definiramo vrednosti $M, m \in \mathbb{R}$, taki, da M navzgor omeji dane vsote, m pa navzdol.

Zdaj lahko zapišemo sledeč linearni program:

$$\begin{aligned}
&\min(M-m)\\
&\forall j \in \{1,\dots,n\}:\ 0 \le v_j \le 1,\ v_j \in \mathbb{Z}\\
&\forall i \in \{1,\dots,k\}:\ \sum_{j=1}^{n} f_j(x_i)v_j \le M\\
&\forall i \in \{1,\dots,k\}:\ \sum_{j=1}^{n} f_j(x_i)v_j \ge m\\
&\sum_{j=1}^{n} v_j \ge 1\\
&\sum_{j=1}^{n} v_j \le 7
\end{aligned}$$

Ker sta M in m realnoštevilski omejitvi, ta linearen program ni celoštevilski ampak mešano-celoštevilski (MCLP).

4 Implementacija

4.1 Generiranje trikotnih funkcij

Definicija trikotne funkcije ni pretirano striktno določena, ideja pa je naslednja. Funkcija je konstantno enaka 0 na intervalu $(-\infty, x_1)$, kjer x_1 predstavlja levi rob trikotnika. Na intervalu $[x_1, x_2]$, kjer je $(x_2, y(x_2))$ vrh trikotnika, je definirana kot linearna funkcija s pozitivnim naklonom, ki potuje skozi točki $(x_1, 0)$ ter $(x_2, y(x_2))$. Na intervalu od x_2 do x_3, kjer je x_3 desni rob trikotnika, je zopet definirana kot linearna funkcija, tokrat z negativnim naklonom ter potuje skozi $(x_2, y(x_2))$ in $(x_3, 0)$. Na intervalu (x_3, ∞) je zopet konstantno enaka 0. Najbolj preprost primer bi bil morda $f(x) = \max\{1 - |x|, 0\}$.

Pri samem generiranju trikotnih funkcij sva se odločila izkoristiti kar najin izbran način spopadanja s linearnim programom, pri katerem funkcijo oceniva le v končnem naboru točk preloma. Zato sva se odločila, da bova funkcije definirala preko njihovih točk preloma. Ideja je preposta: za dobljeno trojico točk $(x_1, 0), (x_2, y_2), (x_3, 0) \in \mathbb{R}^2$, $x_1, x_2, x_3, y_1 \in \mathbb{R}$, $x_1 < x_2 < x_3$, $y_2 > 0$, lahko trikotno funkcijo f definiramo kot

$$f(x) \ = \ \begin{cases} 0\,; & x \in (-\infty, x_1) \cup [x_3, \infty),\\ \frac{y_2}{x_2 - x_1}x - \frac{y_2 x_1}{x_2 - x_1}\,; & x \in [x_1, x_2),\\ \frac{-y_2}{x_3 - x_2}x + \frac{y_2 x_3}{x_3 - x_2}\,; & x \in [x_2, x_3). \end{cases}$$

Točke sva generirala naključno, s pomočjo `Python`-ove knjižnice `random`. Definirala sva razred `TrikotnaFunkcija`, ki generira vrednosti po enakomerni zvezni porazdelitvi,

$$\begin{aligned} x_1, x_2, x_3 \ &\sim \ \mathcal{U}([-2, 2]),\\ y_2 \ &\sim \ \mathcal{U}([-10, 10]), \end{aligned}$$

ter jih nato uredi v točke $(x_1, 0), (x_2, y_2), (x_3, 0)$.

4.2 Koda za MCLP

S pomočjo spodnje kode sva reševala linearni program. Funkcija `vsota_trikotnih_funkcij` reši MCLP. Omenjena funkcija vrne minimalno razliko med M in m, ter trikotne funkcije, izbrane za izračun čim bolj konstantne vsote.

```
def vsota_trikotnih_funkcij(seznam_funkcij, seznam_tock, z):
    k = len(seznam_tock)
    r = len(seznam_funkcij)

    p = MixedIntegerLinearProgram(maximization=False)
    v = p.new_variable(binary=True)
    m = p.new_variable(integer=False, nonnegative=True)
    p.set_objective(m[1]-m[0])

    for x in seznam_tock:
        sum_ = sum(seznam_funkcij[j](x) * v[j] for j in range(r))
        f'{sum_=}'
        f'{sum_ - m[1]}'
        f'{sum_ - m[0]}'
        p.add_constraint(sum_ - m[1], max=0)
        p.add_constraint(sum_ - m[0], min=0)


    sum2_ = sum(v[j] for j in range(z))
    f'{sum2_=}'
    f'{v[0]=}'
    f'{z=}'
    p.add_constraint(sum2_, min=1)
    p.add_constraint(sum2_, max=z)

    vsota = p.solve()
    M, m = p.get_values(m[1]), p.get_values(m[0])
    f'{M=}'
    f'{m=}'
    v_resitve = p.get_values(v)
    f'{p.get_values(v)}'

    vkljucene_funkcije = [seznam_funkcij[j] for j in v_resitve if v_resitve[j]==1]

    return [vsota, vkljucene_funkcije]
```

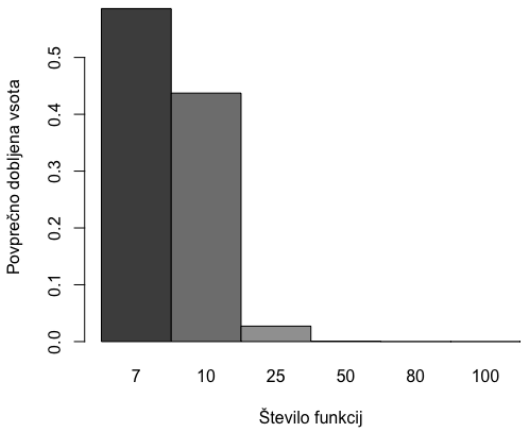
5 Ugotovitve

5.1 Eksperimentiranje

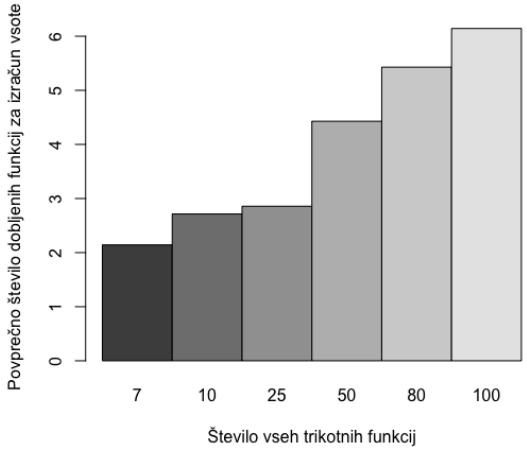
Odločila sva se za generiranje programa tako, da sva spreminjala nabor trikotnih funckij(število n) na intervalu $[0, 1]$. Zanimalo naju je kako konstantna je vosta oziroma kako se spremenja glede na večanje števila n. Predvidevala sva, da večji kot bo n bolj konstantna bo vsota. Odločila sva se, da za vsak n funkcijo `vsota_trikotnih_funkcij` zaženeva desetkrat, saj se nama je to zdelo dovolj. Zanimalo naju je tudi, kako program izbira funckije iz nabora n trikotnih funkcij. Predvidevala sva, da manjše kot bo število n, manjša je verjetnost, da bo izbral večje število trikotnih funkcij za izračun čim bolj konstantne vsote na intervalu $[0, 1]$.

Funckijo `vsota_trikotnih_funkcij` sva torej za vsak n iz nabora $n = 7, 10, 25, 50, 80, 100$ pognala desetkrat. Dobljeno število funkcij za izračun skoraj konstantne vsote trikotnih funkcij na zaprtem itervalu $[0, 1]$ sva zapisovala v seznam in prav tako vsoto. Ugotovila sva, kar je razvidno tudi iz spodnjega grafa (a), da večje kot je število n bolj konstantna je vsota. Večje kot je bilo število n, dlje časa je tudi program porabil za izračun.

Glede na število trikotnih funkcij, ki jih program izbere iz nabora n trikotnih funkcij, sva prišla do ugotovitve, da večje kot je število n, več funkcij je program izbral za izračun, kar je vidno tudi na spodnjem grafu (b).



(a) Skoraj konstantne vsote trikotnih funkcij na zaprtem intervalu

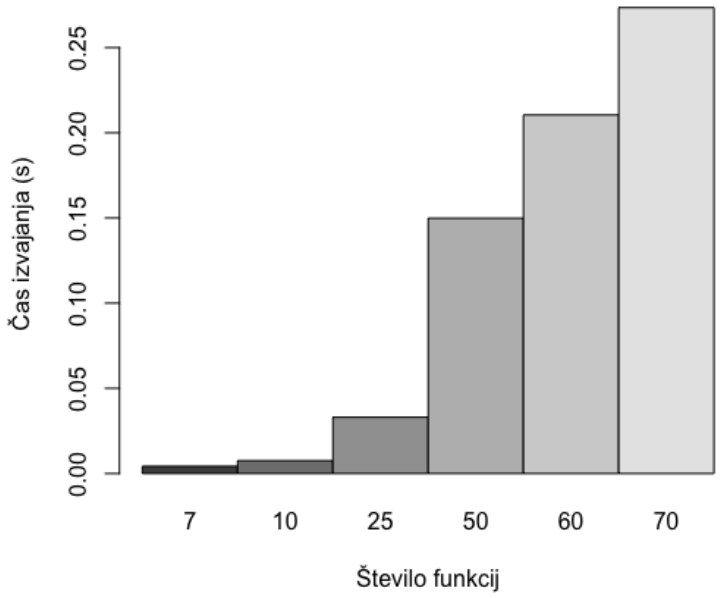


(b) Število izbranih trikotnih funkcij glede na nabor vseh trikotnih funkcij

Še vedno so se pojavili primeri, ko je program izbral po le eno funkcijo iz nabora. Ugotovila sva tudi, da če je bila vsota enaka 0.00, je program ponavaid izbral po eno ali dve funkciji iz nabora n funkcij (seveda je bilo nekaj izjem).

5.2 Čas izvajanja programa

Spodnji graf nam prikazuje, kako se povprečno spreminja čas izvajanja programa, ki je mešano - celoštevilski, za iskanje največ 7 trikotnih funkcij v odvisnosti od celotnega nabora trikonih funkcij. Funckijo `vsota_trikotnih_funkcij` sva pognala za izbor največ 7 trikotnih funkcij izmed n trikotnih funkcij, kjer je $n = 7, 10, 25, 50, 60, 70$. Iz grafa sva razbrala, da se seveda čas izvajanja poveča s povečanjem nabora vseh trikotnih funkcij, kar je precej očitno. Program mora za vsako funkcijo več izračunati vse potrebne vsote in vsako funkcijo izračunati v naboru točk preloma.



Slika 2: Čas izvajanja programa glede na spreminjanje celotnega nabora trikotnih funkcij

Viri

[1] Arjana Žitnik (2021). Prosojnice Celoštevilsko linearno programiranje, časovna zahtevnost in težki problemi. Dostopno tukaj.

[2] Konzultacije z asistentom Janošom Vidalijem in s profesorjem Sergiom Cabellom.