VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Laboratorinis darbas 3

Netiesinis programavimas

Nikita Gainulin

VILNIUS 2024

Turinys

1 Įvadas

Savo ankstesniame laboratoriniame darbe gilinausi į optimizavimą be apribojimų. Kaip jau nustačiau, jo algoritmai be jokių apribojimų nagrinėja visą funkciją ir gali rasti minimumo tašką beveik bet kurioje jos vietoje. Šiame laboratoriniame darbe gilinsiuosi į netiesinį optimizavimą. Vėl susidursime su tam tikrais apribojimais, tačiau šį kartą jie bus pagrįsti tikra, realia problema, kitaip nei vienmačio optimizavimo atveju, kai apribojimai kyla iš pačių algoritmų intervalų pavidalu. Ir kaip ir ankstesnio laboratorinio darbo optimizavimo tipe, toliau veiksime daugiamatėje erdvėje.

2 Nagrinėjama problema

Kaip įprasta, apibrėžkime šiuo metu nagrinėjamą problemą:

Kokia turėtų būti stačiakampio gretasienio formos dėžė, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus?

Pati problema, palyginti su ankstesniu laboratoriniu darbu, nepasikeitė, tačiau optimizavimo, kitaip tariant, sprendimo būdas skirsis. Prieš tęsdami toliau, pažvelkime į išvadą, kurią gavau optimizavęs šį uždavinį be apribojimų: **dėžė turi būti kubas, kurio viena į kitą atgręžtų sienų plotų sumos, kurių yra 3, lygūs $\frac{1}{3}$ paviršiaus vieneto ploto, t.y $2ab = 2bc = 2ac = \frac{1}{3}$, kur a, b, c - atitinkamai ilgis, plotis ir aukštis.**

Nors šis atsakymas ir yra teisingas, ar jis neskambėtų šiek tiek painiai, jei iš tikrųjų bandytume spręsti šią problemą realiame pasaulyje? Ar negalėtume rasti glaustesnio atsakymo, kuriame būtų tiesiai nurodytas konkretus dėžės ilgis, plotis ir aukštis? Netiesinis optimizavimas puikiai tinka tokio tipo problemoms spręsti.

Pirmiausia turėsime iš naujo apibrėžti tikslo funkciją, jei norime gauti daug glaustesnį atsakymą. Tačiau daug transformacijų atlikti nereikės, nes tūrio formulėje ($V = a \cdot b \cdot c$) yra viskas, ko mums reikia:

$$f(X) = -(x_0 \cdot x_1 \cdot x_2) \tag{1}$$

kur x_0 - ilgis a, x_1 - plotis b, x_2 - dėžės aukštis c. Svarbu nepamiršti minuso ženklo, nes, norėdami rasti didžiausią tūrį, turėsime funkciją minimizuoti.

Dabar, kai turime tikslo funkciją, gali kilti klausimas, kodėl pasirinkome būtent netiesinį optimizavimą? Ar negalime rasti dėžės parametrų bet kuriuo kitu optimizavimo tipu? Tiesa, tikrai yra keletas sprendimo būdų, kaip gauti norimus rezultatus, tačiau, kaip minėjau anksčiau, netiesinis optimizavimas puikiai tinka dėl to, kad pačiame uždavinyje jau yra tam tikri apribojimai, kurių turime laikytis. Pažvelkime į juos.

Pirmiausia - lygybinis apribojimas. Pagal užduotį norime rasti didžiausią reikšmę dėžutės ploto vienete, todėl galime sudaryti tokią funkciją $g_i(X)$, kuri užtikrintų, kad mūsų algoritmo pasirinktos reikšmės neišeitų iš ribų:

$$g_i(X) = 2 \cdot (x_0x_1 + x_0x_2 + x_1x_2) - 1 \tag{2}$$

Toliau - nelygybės apribojimas. Kadangi realiame gyvenime dėžutės ilgis, plotis ir aukštis negali būti neigiami, turime tokią nelygybę:

$$x_0, x_1, x_2 \geq 0$$

Pagalvokime apie tai, kaip galime paversti šią nelygybę info funkcija. Pagal bendrąjį standartą žinome, kad funkcija bus tokio pavidalo - $h_i(x) \leq 0$, todėl belieka tik padauginti duotą nelygybę iš

-1:

$$h_i(X) = [-x_0, -x_1, -x_2] \tag{3}$$

Taip Python kalba sukūriau savo pasirinktinę klasę, skirtą šios problemos tikslo (1) ir apribojimų (2), (3) funkcijoms:

```
class Optimization:
    counter = 0

    def __init__(self):
        pass

    def f(self, x):
        Optimization.counter += 1
        return -(x[0]*x[1]*x[2])

    def g(self, x):
        return 2*(x[0]*x[1]+x[0]*x[2]+x[1]*x[2])-1

    def h(self, x):
        return [-x[0], -x[1], -x[2]]

    def reset(self):
        Optimization.counter = 0
```

3 Netiesinio programavimo optimizavimas ir jo algoritmas

4 Rezultatai ir jų analyzė

5 Išvada

6 Priedas