

Optimizacijske metode (IŠRM), 1. domača naloga

ANTON LUKA ŠIJANEC

16. april 2025

Povzetek

Rešitve 1. domače naloge (navodila: <https://ucilnica.fmf.uni-lj.si/mod/resource/view.php?id=73567>).

1. Privzemimo, da delavcev ne moremo odpustiti (glede tega se namreč navodilo ne opredeli). Potemtakem imamo vsak teden možnost nekaj delavcev uporabiti kot učitelje novih delavcev, ostale pa za izdelavo radiev, nobenim pa pravice/dolžnosti do dela ne smemo odvzeti. Teda j moramo zgolj določiti, koliko delavcev bo učiteljev za vsak teden posebej. Očitno ni smiselno uvažati novih delavcev četrti teden, saj nikdar ne bodo imeli priložnosti svojih naučenih veščin izkoristiti. Spremenljivke p_1, p_2, p_3 torej določijo, koliko delavcev bo v posameznem tednu učiteljev, spremenljivke u_1, u_2, u_3 pa določijo, koliko bo v posameznem tednu učencev. Ni namreč nujno, da se vedno izplača popolnoma zapolniti kapaciteto (3) delavca učitelja — učeči se delavci namreč stanejo 100 € / teden.

V prvem tednu bo radie izdelovalo $40 - p_1$ delavcev. V drugem $40 - p_2 + u_1$, ker se bodo v prvem tednu izobraženi delavci pridružili tistim štiridesetim, ki so bili sprva zaposleni, p_2 delavcev pa bo učilo nove delavce, itd.

Seveda morajo biti vsi za svoje delo oziroma uvažanje ustrezno poplačani, torej gre za izvirne delavce $40 \cdot 4 \cdot 200$ evrov, za delavce, pridobljene v prvem tednu $u_1 \cdot 100$ za uvažanje in $3 \cdot u_1 \cdot 200$ pa za nadaljnjo delo, itd.

Linearni program se glasi

$$\begin{aligned} \max & 50(20(40 - p_1) + 18(40 - p_2 + u_1) + 16(40 - p_3 + u_1 + u_2) + 14(40 + u_1 + u_2 + u_3)) \\ & - 40 \cdot 4 \cdot 200 - (u_1 + u_2 + u_3) \cdot 100 - 3 \cdot u_1 \cdot 200 - 2 \cdot u_2 \cdot 200 - u_3 \cdot 200 - 5 \cdot 20000 \\ \text{p. p. } & p_1, p_2, p_3, u_1, u_2, u_3 \geq 0 \\ & p_1 \leq 40, \quad p_2 \leq 40 + u_1, \quad p_3 \leq 40 + u_1 + u_2 \\ & u_1 \leq 3 \cdot p_1, \quad u_2 \leq 3 \cdot p_2, \quad u_3 \leq 3 \cdot p_3 \\ & 50(40 - p_1 + 40 - p_2 + u_1 + 40 - p_3 + u_1 + u_2 + 40 + u_1 + u_2 + u_3) = 20000. \end{aligned}$$

Imamo še eno dodatno omejitev; p_1, p_2 in p_3 morajo biti cela števila. Nepredstavljivo težko je namreč zaposliti zgolj nek neceli delež študenta. Problem rešimo s pythonsko knjižnico Sage Math[1]:

```
#!/usr/bin/sage
```

```
lp = MixedIntegerLinearProgram()
p = lp.new_variable(real=True, nonnegative=True)
p1, p2, p3 = p['1'], p['2'], p['3']
u = lp.new_variable(real=True, nonnegative=True)
u1, u2, u3 = u['1'], u['2'], u['3']
lp.set_objective(50*(20*(40-p1)+18*(40-p2+u1)+16*(40-p3+u1+u2)+14*(40+u1+u2+u3))
                -40*4*200-(u1+u2+u3)*100-3*u1*200-2*u2*200-u3*200-5*20000)
lp.add_constraint(p1 >= 0)
lp.add_constraint(p2 >= 0)
lp.add_constraint(p3 >= 0)
lp.add_constraint(u1 >= 0)
lp.add_constraint(u2 >= 0)
lp.add_constraint(u3 >= 0)
lp.add_constraint(p1 <= 40)
```

```

lp.add_constraint(p2 <= 40+u1)
lp.add_constraint(p3 <= 40+u1+u2)
lp.add_constraint(u1 <= p1*3)
lp.add_constraint(u2 <= p2*3)
lp.add_constraint(u3 <= p3*3)
lp.add_constraint(50*(40-p1+40-p2+u1+40-p3+u1+u2+40+u1+u2+u3) == 20000)
print("opt_vrednost:_" + str(lp.solve()))
print("p1=" + str(lp.get_values(p1)) + ",_p2=" + str(lp.get_values(p2)) + ",_p3="
      + str(lp.get_values(p3)) + ",_u1=" + str(lp.get_values(u1)) + ",_u2=" +
      str(lp.get_values(u2)) + ",_u3=" + str(lp.get_values(u3)))

```

Izhod programa:

```

opt_vrednost: 127000.0
p1=30.0, p2=0.0, p3=0.0, u1=90.0, u2=0.0, u3=0.0

```

Prvi teden naj torej podjetje izvežba devetdeset novih delavcev, druge tedne pa naj ne uvaja novih delavcev.

Kaj pa, če predpostavko z začetka rešitve spremenimo? Za trenutek odmislimo delavske pravice; kako se spremeni optimalna rešitev, če se lahko podjetje vsak teden sproti odloči, koliko delavcev bo delalo vsak teden? Dodajmo še spremenljivke o_1, o_2, o_3, o_4 , ki povedo, koliko delavcev bo vsak teden delalo in popravimo linearni program:

$$\begin{aligned}
& \max 50(20o_1 + 18o_2 + 16o_3 + 14o_4) \\
& - (u_1 + u_2 + u_3) \cdot 100 - 200 \cdot (o_1 + o_2 + o_3 + o_4 + p_1 + p_2 + p_3) - 5 \cdot 20000 \\
& p, p, p_1, p_2, p_3, u_1, u_2, u_3, o_1, o_2, o_3, o_4 \geq 0 \\
& p_1 \leq 40, \quad p_2 \leq 40 + u_1, \quad p_3 \leq 40 + u_1 + u_2 \\
& u_1 \leq 3 \cdot p_1, \quad u_2 \leq 3 \cdot p_2, \quad u_3 \leq 3 \cdot p_3 \\
& o_1 \leq 40 - p_1 \quad o_2 \leq 40 - p_2 + u_1, \quad o_3 \leq 40 - p_3 + u_1 + u_2, \quad o_4 = 40 + u_1 + u_2 + u_3 \\
& 50(o_1 + o_2 + o_3 + o_4) = 20000.
\end{aligned}$$

```

#!/usr/bin/sage
lp = MixedIntegerLinearProgram()
p = lp.new_variable(real=True, nonnegative=True)
p1, p2, p3 = p['1'], p['2'], p['3']
u = lp.new_variable(real=True, nonnegative=True)
u1, u2, u3 = u['1'], u['2'], u['3']
o = lp.new_variable(real=True, nonnegative=True)
o1, o2, o3, o4 = o['1'], o['2'], o['3'], o['4']
lp.set_objective(50*(20*o1+18*o2+16*o3+14*o4)-(u1+u2+u3)*100-(o1+o2+o3+o4+p1+p2
+p3)*200-5*20000)
lp.add_constraint(p1 >= 0)
lp.add_constraint(p2 >= 0)
lp.add_constraint(p3 >= 0)
lp.add_constraint(u1 >= 0)
lp.add_constraint(u2 >= 0)
lp.add_constraint(u3 >= 0)
lp.add_constraint(p1 <= 40)
lp.add_constraint(p2 <= 40+u1)
lp.add_constraint(p3 <= 40+u1+u2)
lp.add_constraint(u1 <= p1*3)
lp.add_constraint(u2 <= p2*3)
lp.add_constraint(u3 <= p3*3)
lp.add_constraint(o1 <= 40-p1)
lp.add_constraint(o2 <= 40-p2+u1)
lp.add_constraint(o3 <= 40-p3+u1+u2)

```

```

lp.add_constraint(o4 <= 40+u1+u2+u3)
lp.add_constraint(50*(o1+o2+o3+o4) == 20000)
print("opt_vrednost:_" + str(lp.solve()))
print("p1=" + str(lp.get_values(p1)) + ",_p2=" + str(lp.get_values(p2)) + ",_p3="
      + str(lp.get_values(p3)) + ",_u1=" + str(lp.get_values(u1)) + ",_u2=" +
      str(lp.get_values(u2)) + ",_u3=" + str(lp.get_values(u3)) + ",_o1=" + str(
      lp.get_values(o1)) + ",_o2=" + str(lp.get_values(o2)) + ",_o3=" + str(lp.
      get_values(o3)) + ",_o4=" + str(lp.get_values(o4)))

```

Izhod programa:

```

opt_vrednost: 128000.0
p1=40.0, p2=0.0, p3=0.0, u1=120.0, u2=0.0, u3=0.0, o1=0.0, o2=160.0, o3=160.0, o4=80.0

```

Komentar: če delavcem podjetje ne zagotavlja redne službe, ima večji dobiček!

- Možnosti igralcev so pari $(i \in \{2, 3, 4\}, f : \{2, 3, 4\} \rightarrow \{-1, 0, 1\})$, kjer je i izbrano število igralca, f pa funkcija, ki pove, kako bo igralec popravil svojo število glede na število, ki ga je izbral soigralec. Takih funkcij je $3^3 = 27$ (oštevilčimo jih), torej je možnosti $3 \cdot 3^3 = 81$. Plačilna matrika matrične igre bo torej res dimenzij 81×81 .

V Pythonu izdelamo iterator vseh možnosti, iteriramo čez vse možne pare možnosti (dva igralca) in izpolnimo Numpyjevsko tabelo, ki služi kot plačilna matrika. S knjižnico Nashpy[2] na igri, ki ji pripada ta matrika, najdemo optimalni strategiji za oba igralca in vrednost igre.

```

#!/usr/bin/python3
import itertools
import fractions
import nashpy
import numpy
b = [
    [0, 2, 0, 1, -4],
    [-2, 0, -3, 1, 0],
    [0, -3, 0, -1, -2],
    [-1, -1, 1, 0, 2],
    [4, 0, 2, -2, 0]]
izbire = [2, 3, 4]
poprave = [-1, 0, 1]
m = numpy.full((81, 81), 0xdeadbeef)
def možnosti():
    return itertools.product(range(len(izbire)), itertools.product(poprave,
        repeat=3))
for (index_prvi, poteza_prvi), (index_drugi, poteza_drugi) in itertools.product(
    enumerate(možnosti()), repeat=2):
    prvi_končno = izbire[poteza_prvi[0]] + poteza_prvi[1][poteza_drugi[0]]
    drugi_končno = izbire[poteza_drugi[0]] + poteza_drugi[1][poteza_prvi
        [0]]
    plača = b[prvi_končno-1][drugi_končno-1]
    m[index_prvi][index_drugi] = plača
# numpy.set_printoptions(threshold=numpy.inf, linewidth=69420) # print(m)
if numpy.any(numpy.isin(m, 0xdeadbeef)):
    raise Exception("some_unset_values")
g = nashpy.Game(m)
opt = g.linear_program()
razlika = g[opt][0]+g[opt][1]
if razlika != 0:
    raise Exception("nisem_našel_optimalne_strategije._razlika_" + str(
        razlika))
for i in range(2):
    print("opt_strategija_%d._igralca:\n" % (i + 1) + ",\n".join(["p" +
        str(list(možnosti())[idx]) + "=" + str(fractions.Fraction(x).

```

```

        limit_denominator(0xffffffff)) for idx, x in enumerate(opt[i]) if x
        != 0]), "\n")
print("vrednost igre: ", fractions.Fraction(g[opt][0]).limit_denominator(0
    xffffffff))

```

Izhod programa:

```

opt. strategija 1. igralka:
p(0, (-1, -1, 1))=3/13,
p(0, (1, -1, 0))=2/13,
p(1, (1, -1, 1))=2/13,
p(1, (1, 1, -1))=1/13,
p(1, (1, 1, 1))=2/13,
p(2, (1, 0, 0))=2/13,
p(2, (1, 1, 0))=1/13

```

```

opt. strategija 2. igralka:
p(0, (1, -1, 0))=8/117,
p(0, (1, 0, 0))=4/117,
p(1, (0, -1, -1))=5/13,
p(1, (0, -1, 1))=1/13,
p(1, (0, 0, 1))=3/26,
p(2, (-1, -1, 0))=5/78,
p(2, (-1, 0, 0))=5/78,
p(2, (1, 0, 0))=5/26

```

vrednost igre: $-5/13$

Ker vrednost igre ni 0, igra ni poštena. Navodili za igralka:

- Prvi igralec naj
 - v $3/13$ iger izbere 0 in $f(2) = -1, f(3) = -1, f(4) = 1$
 - v $2/13$ iger izbere 0 in $f(2) = 1, f(3) = -1, f(4) = 0$
 - v $2/13$ iger izbere 1 in $f(2) = 1, f(3) = -1, f(4) = 1$
 - v $1/13$ iger izbere 1 in $f(2) = 1, f(3) = 1, f(4) = -1$
 - v $2/13$ iger izbere 1 in $f(2) = 1, f(3) = 1, f(4) = 1$
 - v $2/13$ iger izbere 2 in $f(2) = 1, f(3) = 0, f(4) = 0$
 - v $1/13$ iger izbere 2 in $f(2) = 1, f(3) = 1, f(4) = 0$
- Drugi igralec naj
 - v $8/117$ iger izbere 0 in $f(2) = 1, f(3) = -1, f(4) = 0$
 - v $4/117$ iger izbere 0 in $f(2) = 1, f(3) = 0, f(4) = 0$
 - v $5/13$ iger izbere 1 in $f(2) = 0, f(3) = -1, f(4) = -1$
 - v $1/13$ iger izbere 1 in $f(2) = 0, f(3) = -1, f(4) = 1$
 - v $3/26$ iger izbere 1 in $f(2) = 0, f(3) = 0, f(4) = 1$
 - v $5/78$ iger izbere 2 in $f(2) = -1, f(3) = -1, f(4) = 0$
 - v $5/78$ iger izbere 2 in $f(2) = -1, f(3) = 0, f(4) = 0$
 - v $5/26$ iger izbere 2 in $f(2) = 1, f(3) = 0, f(4) = 0$

Matrika se nahaja spodaj.

Literatura

- [1] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 10.6)*, 2025, <https://www.sagemath.org>.

7