Laboratorijska vježba 3

Tim:

Amina Babić 18492

Elma Šeremet

Aldin Alić

Linearno programiranje u Julia jeziku

1. Korištenjem JuMP[1] paketa i GLPK solver-a za problema linearnog programiranja, riješiti tri različito postavljena problema s predavanja.
2. Za vježbu sastavite matematske modele i riješite, korištenjem prethodno navedenog alata, tri problema iz dokumenta Zadaci za samostalno vježbanje iz linearnog programiranja.

U narednim primjerima / zadacima podrazumijeva se korištenje paketa JuMP i GLPK.

ZADATAK 1.

PRIMJER 1.

Primjer : Planira se proizvodnja tri tipa detrdženta D_1, D_2 i D_3. Sa trgovačkom mrežom je dogovorena isporuka tačno 100 kg detrdženta bez obzira na tip. Za uvoz odgovarajućeg repromaterijala planirano su sredstva u iznosu od 110 $. Po jednom kilogramu detrdženta, za proizvodnju detrdženata D_1, D_2 i D_3 treba nabaviti repromaterijala u vrijednosti 2 $, 1.5 $ odnosno 0.5 $. Također je planirano da se za proizvodnju uposle radnici sa angažmanom od ukupno barem 120 radnih sati, pri čemu je za proizvodnju jednog kilograma detrdženata D_1, D_2 i D_3 potrebno uložiti respektivno 2 sata, 1 sat odnosno 1 sat. Prodajna cijena detrdženata D_1, D_2 i D_3 po kilogramu respektivno iznosi 10 KM, 5 KM odnosno 8 KM. Formirati matematski model iz kojeg se može odrediti *koliko treba proizvesti* svakog od tipova detrdženata da se pri tome ostvari maksimalna moguća zarada.

```
function primjer1()

    m = Model(GLPK.Optimizer)

    @variable(m, x1 >= 0)
    @variable(m, x2 >= 0)

    @objective(m, Max, 2x1 - 3x2 + 800)

    @constraint(m, constraint1, 1.5x1 + x2 <= 60)
    @constraint(m, constraint2, x1 >= 20)
    @constraint(m, constraint3, x1 + x2 <= 100)

    print("MODEL:\n\n", m)

    optimize!(m)

    println("Termination status: " * string(termination_status(m)))
    println("Constraints values: ")
    println(value(constraint1))
    println(value(constraint2))
    println(value(constraint3))

    s =
        "Rješenja su: " *
        "\n x1 = " * string(value(x1)) *
        "\n x2 = " * string(value(x2)) *
        "\nVrijednost cilja: " * string(objective_value(m))
```

```
MODEL:

Max 2 x1 - 3 x2 + 800
Subject to
 constraint2 : x1 ≥ 20.0
 constraint1 : 1.5 x1 + x2 ≤ 60.0
 constraint3 : x1 + x2 ≤ 100.0
 x1 ≥ 0.0
 x2 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
60.0
40.0
40.0
Rješenja su:
 x1 = 40.0
 x2 = 0.0
Vrijednost cilja: 880.0
```

```
    println(s)
end
primjer1()
```

PRIMJER 2.

Primjer : Teretni brod prebacuje dvije vrste tereta. Zbog prirode tereta, za svaku utovarenu tonu prve vrste tereta brodovlasnik treba platiti taksu od 500 KM, dok za svaku utovarenu tonu druge vrste tereta brodovlasnik dobija "kaparu" (predujam) od 200 KM. Gotovina kojim brodovlasnik raspolaže na početku utovara iznosi 3000 KM i brodovlasnik nema načina da u trenutku utovara nabavi veću količinu novca. Naravno, novac koji se dobije kao predujam zbog utovara druge vrste tereta može se koristiti za plaćanje takse za prvu vrstu tereta. Utovar prve vrste tereta traje pola sata po toni, a druge vrste tereta 15 minuta po toni. Ukupno vrijeme koje je na raspolaganju za utovar je najviše 12 sati, a u jednom trenutku može se utovarati samo jedna vrsta tereta. Zarada od prevoza prve vrste tereta iznosi 2000 KM po toni, a od drugog 100 KM po toni, koja se isplaćuje po obavljenom prevozu (u ovaj iznos nije uračunata ranije spomenuta "kapara"). Formirati matematski model iz kojeg se može odrediti koliko treba utovariti svake vrste tereta tako da se njegovim transportom ostvari maksimalna zarada.

```
function primjer2()

    m = Model(GLPK.Optimizer)

    @variable(m, x1 >= 0)
    @variable(m, x2 >= 0)

    @objective(m, Max, 3x1 + 2x2)

    @constraint(m, constraint1, 0.5x1 + 0.25x2 <= 12)
    @constraint(m, constraint2, 500x1 - 200x2 <= 3000)

    print("\nMODEL:\n\n", m)

    optimize!(m)
    println("Termination status: " * string(termination_status(m)))

    println("Constraints values: ")
    println(value(constraint1))
    println(value(constraint2))

    s =
        "Rješenja su: " *
        "\n x1 = " *
        string(value(x1)) *
        "\n x2 = " *
        string(value(x2)) *
        "\nVrijednost cilja: " *
        string(objective_value(m))

    println(s)
```

```
MODEL:

Max 3 x1 + 2 x2
Subject to
 constraint1 : 0.5 x1 + 0.25 x2 ≤ 12.0
 constraint2 : 500 x1 - 200 x2 ≤ 3000.0
 x1 ≥ 0.0
 x2 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
12.0
-9600.0
Rješenja su:
 x1 = 0.0
 x2 = 48.0
Vrijednost cilja: 96.0
```

```
end
primjer2()
```

PRIMJER 3.

Primjer: Potrebno je obezbijediti vitaminsku terapiju koja će sadržavati četiri vrste vitamina V_1, V_2, V_3 i V_4. Na raspolaganju su dvije vrste vitaminskih sirupa S_1 i S_2 čije su cijene 40 n.j./g i 30 n.j./g respektivno. Vitaminski koktel mora sadržavati najmanje 0.2 g, 0.3 g, 3 g i 1.2 g vitamina V_1, V_2, V_3 i V_4 respektivno. Sljedeća tabela pokazuje sastav pojedinih vitamina u obje vrste vitaminskih sirupa:

	V_1	V_2	V_3	V_4
S_1	10 %	0 %	50 %	10 %
S_2	0 %	10 %	30 %	20 %

Formirati matematski model iz kojeg se može odrediti *koliko treba nabaviti* sirupa S_1 a koliko sirupa S_2 tako da ukupni trošak bude minimalan.

```
function primjer3()

    m = Model(GLPK.Optimizer)

    @variable(m, x1 >= 0)
    @variable(m, x2 >= 0)

    @objective(m, Min, 40x1 + 30x2)

    @constraint(m, constraint1, 0.1x1 >= 0.2)
    @constraint(m, constraint2, 0.1x2 >= 0.3)
    @constraint(m, constraint3, 0.5x1 + 0.3x2 >=3)
    @constraint(m, constraint4, 0.1x1 + 0.2x2 >= 1.2)

    print("\nMODEL:\n\n", m)

    optimize!(m)
    println("Termination status: " * string(termination_status(m)))

    println("Constraints values: ")
    println(value(constraint1))
    println(value(constraint2))
    println(value(constraint3))
    println(value(constraint4))

    s =
        "Rješenja su: " *
        "\n x1 = " *
        string(value(x1)) *
        "\n x2 = " *
```

```
MODEL:

Min 40 x1 + 30 x2
Subject to
 constraint1 : 0.1 x1 ≥ 0.2
 constraint2 : 0.1 x2 ≥ 0.3
 constraint3 : 0.5 x1 + 0.3 x2 ≥ 3.0
 constraint4 : 0.1 x1 + 0.2 x2 ≥ 1.2
 x1 ≥ 0.0
 x2 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
0.34285714285714297
0.4285714285714285
3.0
1.2
Rješenja su:
 x1 = 3.4285714285714297
 x2 = 4.285714285714285
Vrijednost cilja: 265.7142857142858
```

```
    string(value(x2)) *
    "\nVrijednost cilja: " *
    string(objective_value(m))

  println(s)
end
primjer3()
```

ZADATAK 2.

Kompanija za proizvodnju slatkiša proizvodi visokokvalitetne čokoladne proizvode i namjerava pokrenuti proizvodnju dva nova slatkiša. Proizvodi se prave u tri različita odjeljka u kojem provode određeno vrijeme. Prvi proizvod zahtijeva 1 h proizvodnje u odjeljku 1 i 3 h proizvodnje u odjeljku 3 po jednom komadu. Drugi proizvod zahtijeva 1 h proizvodnje u odjeljku 2 i 2 h proizvodnje u odjeljku 3 po jedom komadu. Odjeljak 1 ima na raspolaganju 3 slobodna sata, odjeljak 2 ima 6 slobodnih sati i odjeljak 3 ima 18 slobodnih sati. Svi proizvedeni novi proizvodi mogu se prodati a cijena prvog iznosi 2 KM, a drugog 4 KM po komadu.

Rješenje:

- Potrebno je proizvesti 2 komada prvog i 6 komada drugog slatkiša, pri čemu će se ostvariti zarada od 28 KM.

$$\arg\max Z = 2x_1 + 4x_2$$

p.o.

$$x_1 >= 0,\ x_2 >= 0$$
$$x_1 <= 3$$
$$x_2 <= 6$$
$$3x_1 + 2x_2 <= 18$$

```
MODEL:

Max 2 x1 + 4 x2
Subject to
 constraint1 : x1 ≤ 3.0
 constraint2 : x2 ≤ 6.0
 constraint3 : 3 x1 + 2 x2 ≤ 18.0
 x1 ≥ 0.0
 x2 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
2.0
6.0
Rješenja su:
 x1 = 2.0
 x2 = 6.0
Vrijednost cilja: 28.0
```

```
function zadatak1()

  m = Model(GLPK.Optimizer)

  @variable(m, x1 >= 0)
  @variable(m, x2 >= 0)

  @objective(m, Max, 2x1 + 4x2)

  @constraint(m, constraint1, x1 <= 3)
  @constraint(m, constraint2, x2 <= 6)
  @constraint(m, constraint3, 3x1 + 2x2 <=18)

  print("\nMODEL:\n\n", m)
  optimize!(m)
  println("Termination status: " * string(termination_status(m)))

  println("Constraints values: ")
  println(value(constraint1))
  println(value(constraint2))

  s =
```

```
    "Rješenja su: " *
    "\n x1 = " *
    string(value(x1)) *
    "\n x2 = " *
    string(value(x2)) *
    "\nVrijednost cilja: " *
    string(objective_value(m))
  println(s)
end
zadatak1()
```

Neka fabrika proizvodi geveznice i kataklingere, pri čemu je prodajna cijena geveznica 150 KM po kubnom metru, a kataklingera 40 KM po kilogramu. Za proizvodnju jednog kubnog metra geveznica potrebna su 3 kilograma cincozni i 9 vrećica šnaus-mufni, dok su za proizvodnju jednog kilograma kataklingera potrebna 2 litra kalamute i 4 vrećice šnaus-mufni. Fabrika raspolaže zalihama od 36 kilograma cincozni, 54 litara kalamute i 144 vrećice šnaus-mufni. Potrebno je naći optimalni plan proizvodnje koji će maksimizirati moguću zaradu koja će se ostvariti prodajom, u skladu sa raspoloživim zalihama.

Rješenje:

– Potrebno je proizvesti 12 m^3 geveznica i 9 kg kataklingera, pri čemu će se ostvariti zarada od 2160 KM.

arg max Z = 150x1 + 40x2
p.o.
3x1 <= 36
2x2 <= 54
9x1 + 4x2 <= 144
x1 >= 0, x2 >= 0

```
MODEL:

Max 150 x1 + 40 x2
Subject to
 constraint1 : 3 x1 ≤ 36.0
 constraint2 : 2 x2 ≤ 54.0
 constraint3 : 9 x1 + 4 x2 ≤ 144.0
 x1 ≥ 0.0
 x2 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
36.0
18.0
Rješenja su:
 x1 = 12.0
 x2 = 9.0
Vrijednost cilja: 2160.0
```

```
function zadatak2()

  m = Model(GLPK.Optimizer)

  @variable(m, x1 >= 0)
  @variable(m, x2 >= 0)

  @objective(m, Max, 150x1 + 40x2)

  @constraint(m, constraint1, 3x1 <= 36)
  @constraint(m, constraint2, 2x2 <= 54)
  @constraint(m, constraint3, 9x1 + 4x2 <=144)

  print("\nMODEL:\n\n", m)

  optimize!(m)
  println("Termination status: " * string(termination_status(m)))

  println("Constraints values: ")
  println(value(constraint1))
  println(value(constraint2))

  s =
    "Rješenja su: " *
```

```
        "\n x1 = " *
        string(value(x1)) *
        "\n x2 = " *
        string(value(x2)) *
        "\nVrijednost cilja: " *
        string(objective_value(m))

    println(s)
end
zadatak2()
```

Fabrika može proizvoditi tri proizvoda P_1, P_2 i P_3, pri čemu se koriste tri sirovine S_1, S_2 i S_3. Za proizvodnju prvog proizvoda koriste se dvije količinske jedinice prve i tri količinske jedinice druge sirovine. Za proizvodnju drugog proizvoda koriste se dvije količinske jedinice prve, tri količinske jedinice druge i jedna količinska jedinica treće sirovine. Za proizvodnju trećeg proizvoda potrebno je dvije količinske jedinice prve sirovine i jedna količinska jedinica treće sirovine. Dobit od jedne količinske jedinice prvog proizvoda je dvije novčane jedinice, od drugog tri novčane jedinice, a od trećeg jedna novčana jedinica. Ako su količine sirovina za planski period ograničene na četiri količinske jedinice za prvu sirovinu, dvije za drugu i tri za treću, potrebno je napraviti optimalni plan proizvodnje koji de uz zadana ograničenja ostvariti najveću novčanu dobit.

Rješenje:

– Proizvod P_1, ne treba proizvoditi, a treba prozivoditi 2/3 odnosno 4/3 količinskih jedinica proizvoda P_2 odnosno P_3, pri čemu će ukupno ostvarena zarada iznositi 10/3 novčanih jedinica.

```
function zadatak3()

    m = Model(GLPK.Optimizer)

    @variable(m, x1 >= 0)
    @variable(m, x2 >= 0)
    @variable(m, x3 >= 0)

    @objective(m, Max, 2x1 + 3x2 + x3)

    @constraint(m, constraint1, 2x1 + 2x2 + 2x3 <= 4)
    @constraint(m, constraint2, 3x1 + 3x2 <= 2)
    @constraint(m, constraint3, x2 + x3 <= 3)

    print("\nMODEL:\n\n", m)

    optimize!(m)
    println("Termination status: " *
string(termination_status(m)))

    println("Constraints values: ")
    println(value(constraint1))
    println(value(constraint2))
    println(value(constraint3))

    s =
        "Rješenja su: " *
        "\n x1 = " *
        string(value(x1)) *
```

```
MODEL:

Max 2 x1 + 3 x2 + x3
Subject to
 constraint1 : 2 x1 + 2 x2 + 2 x3 ≤ 4.0
 constraint2 : 3 x1 + 3 x2 ≤ 2.0
 constraint3 : x2 + x3 ≤ 3.0
 x1 ≥ 0.0
 x2 ≥ 0.0
 x3 ≥ 0.0
Termination status: OPTIMAL
Constraints values:
4.0
2.0
2.0
Rješenja su:
 x1 = 0.0
 x2 = 0.6666666666666666
 x3 = 1.3333333333333335
Vrijednost cilja: 3.3333333333333335
```

```
        "\n x2 = " *
        string(value(x2)) *
        "\n x3 = " *
        string(value(x3)) *
        "\nVrijednost cilja: " *
        string(objective_value(m))

    println(s)
end
zadatak3()
```