Semestrální práce č. 3

Operační analýza

Jan Burian

1. Zadání

Zápočtová práce 4

Jméno a přijmení: Jan

Fakulta, ročník: FAV4

Maticová hra

Zadaní:

Najděte řešení maticové hry

4	9	5	10
9	10	8	6
6	8	5	7
10	7	3	2
4	10	4	2

v oblasti rozšířených strategií.

Zadání vygenerované systémem "OA2000"

2. Vypracování

2.1 Nalezení rovnovážného řešení

$$M = \begin{bmatrix} 4 & 9 & 5 & 10 \\ 9 & 10 & 8 & 6 \\ 6 & 8 & 5 & 7 \\ 10 & 7 & 3 & 2 \\ 4 & 10 & 4 & 2 \end{bmatrix}$$

Maximin kritérium

				min	max
4	9	5	10	4	
9	10	8	**6**	6	6
6	8	**5**	7	5	
10	**7**	3	**2**	2	
4	10	4	**2**	2	

Minimax kritérium

	4	9	5	**10**
	9	**10**	**8**	6
	6	8	5	7
	10	7	3	2
	4	10	4	2
max	10	10	8	10
min			8	

Sedlový bod (rovnovážné řešení) v tomto případě neexistuje, jelikož se největší z řádkových minim nerovná s nejmenším sloupcovým maximem.

Úloha bude řešena pomocí lineárního programování - bude třeba vypočítat primární i duální úlohu lineárního programování.

2.2 Formulace úlohy

V prvním případě hráč 1 maximalizuje minimální výhru *h* (maximin kritérium), z čehož vyplývá, že minimalizuje $z = \frac{1}{h}$. Cílová funkce bude tedy definována následovně:

$$z = x_1 + x_2 + x_3 + x_4 + x_5 \longrightarrow \text{min.}$$

Současně budou platit následující omezení:

$$Mx \geq 1,$$
$$x \geq 0,$$
$$M > 0$$

V tomto případě se jedná o duální úlohu lineárního programování. Optimální smíšené strategie hráče 1, lze získat použitím následujícího vztahu:

$$\alpha^* = hx^*,$$

kde α^* je optimální strategie hráče 1, h je cena hry a x^* je řešením duální úlohy. Cenu hry lze definovat následovně:

$$h = \frac{1}{z^*},$$

kde z^* je výsledná hodnota kriteriální funkce.

Ve druhém případě hráč 2 minimalizuje maximální výhru h (minimax kritérium), z čehož vyplývá, že maximalizuje $z = \frac{1}{h}$. Cílová funkce bude tedy definována následovně:

$$z = y_1 + y_2 + y_3 + y_4 \longrightarrow \max.$$

Současně budou platit následující omezení:

$$My \leq 1,$$
$$y \geq 0,$$
$$M > 0$$

V tomto případě se jedná o primární úlohu lineárního programování. Optimální smíšené strategie hráče 2, lze získat použitím následujícího vztahu:

$$\beta^* = hy^*,$$

kde β^* je optimální strategie hráče 2, h je cena hry a y^* je řešením primární úlohy. Cenu hry lze definovat následovně:

$$h = \frac{1}{z^*},$$

kde z^* je výsledná hodnota kriteriální funkce.

2.3 Omezující podmínky (pro úlohu maximalizace)

$4y_1 + 9y_2 + 5y_3 + 10y_4 \leq 1$

$9y_1 + 10y_2 + 8y_3 + 6y_4 \leq 1$

$6y_1 + 8y_2 + 5y_3 + 7y_4 \leq 1$

$10y_1 + 7y_2 + 3y_3 + 2y_4 \leq 1$

$4y_1 + 10y_2 + 4y_3 + 2y_4 \leq 1$

2.4 Omezující podmínky (pro úlohu minimalizace)

$4x_1 + 9x_2 + 6x_3 + 10x_4 + 4x_5 \geq 1$

$9x_1 + 10x_2 + 8x_3 + 7x_4 + 10x_5 \geq 1$

$5x_1 + 8x_2 + 5x_3 + 3x_4 + 4x_5 \geq 1$

$10x_1 + 6x_2 + 7x_3 + 2x_4 + 2x_5 \geq 1$

2.5 Definice modelu pomocí matic (maximalizace)

Úlohu je možné obecně zapsat ve tvaru:

$$z = cy$$

$$My \leq b.$$

V našem případě pro :

$$c = [c_i] = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$M = [m_{ij}] = \begin{bmatrix} 4 & 9 & 5 & 10 \\ 9 & 10 & 8 & 6 \\ 6 & 8 & 5 & 7 \\ 10 & 7 & 3 & 2 \\ 4 & 10 & 4 & 2 \end{bmatrix}$$

$$b = [b_j] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

2.6 Definice modelu pomocí matic (minimalizace)

Úlohu je možné obecně zapsat ve tvaru:

$$z = cx$$

$$Mx \geq b.$$

V našem případě pro :

$$c = [c_i] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$M = [m_{ij}] = \begin{bmatrix} 4 & 9 & 6 & 10 & 4 \\ 9 & 10 & 8 & 7 & 10 \\ 5 & 8 & 5 & 3 & 4 \\ 10 & 6 & 7 & 2 & 2 \end{bmatrix}$$

$$b = [b_j] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Matice M bylo třeba v tomto případě transponovat, proto aby bylo možné získat hodnoty vektoru x. Z toho vyplývají změny rozměrů vektorů c a b.

2.7 Příprava nástroje

Instalace knihovny OR-Tools od Googlu

```
!pip install ortools
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
```

```
Requirement already satisfied: ortools in /usr/local/lib/python3.7/dist-packages (9.
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: absl-py>=0.13 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: protobuf>=3.19.4 in /usr/local/lib/python3.7/dist-pac
```

Import potřebného "solveru" pro lineární programování (`linear_solver`)

```
from ortools.linear_solver import pywraplp
```

Solver je nyní nutné inicializovat. Pro potřeby této semestrální práce bude použit Google's linear programming system (GLOP).

```
solver = pywraplp.Solver.CreateSolver('GLOP')
```

2.8 Řešení úlohy

2.8.1 Potřebné metody

```
# Definice proměnných
def define_variables(solver, c, lower_limits):
    x = {}
    n_vars = len(c)
    for j in range(n_vars):
        x[j] = solver.NumVar(lower_limits[j], solver.infinity(), 'x{}'.format(j+1))
    return x, n_vars

# Definice omezujících podmínek
def set_constraints(solver, x, A, b, typ):
    n_vars = len(x)
    n_constraints = len(b)
    constraint = 0
    for i in range(n_constraints):
      if typ == 'max':
        constraint = solver.RowConstraint(-solver.infinity(), b[i], '') # maximalizace
      elif typ == 'min':
        constraint = solver.RowConstraint(b[i], solver.infinity(), '') # minimalizace

      for j in range(n_vars):
          constraint.SetCoefficient(x[j], A[i][j])
    return n_constraints

# Definice cílové funkce
def set_objective(solver, x, c, opt_type='max'):
    n_vars = len(x)
    objective = solver.Objective()
    for j in range(n_vars):
        objective.SetCoefficient(x[j], c[j])
```

```
    if opt_type == 'max':
        objective.SetMaximization()
    elif opt_type == 'min':
        objective.SetMinimization()
    else:
        raise TypeError("Typ optimalizace '{}' není podporován".format(opt_type))
```

2.8.2 Zadání úlohy

```
# Úloha maximalizace
A_max = [[4, 9, 5, 10],        # definice koeficientů omezujících podmínek
    [9, 10, 8, 6],
    [6, 8, 5, 7],
    [10, 7, 3, 2],
    [4, 10, 4, 2]]
b_max = [1, 1, 1, 1, 1]       # definice omezení # maximalizace
c_max = [1, 1, 1, 1]        # definice koeficientů cílové funkce # maximalizace

# Úloha minimalizace
A_min = [[4, 9, 6, 10, 4],        # Transponovat pro min
    [9, 10, 8, 7, 10],
    [5, 8, 5, 3, 4],
    [10, 6, 7, 2, 2]]

b_min = [1, 1, 1, 1]           # definice omezení # minimalizace
c_min = [1, 1, 1, 1, 1] # definice koeficientů cílové funkce # minimalizace
```

2.8.3 Úloha maximalizace

```
lower_limits_max = [0, 0, 0, 0] # max
x_max, n_vars_max = define_variables(solver, c_max, lower_limits_max)     # definice promě
n_constraints_max = set_constraints(solver, x_max, A_max, b_max, 'max')    # nastavení ome
set_objective(solver, x_max, c_max, 'max')            # nastavení cílové funkce

solver.Solve()
```

```
0
```

Výpis výsledků (maximalizace):

```
print('Počet proměnných = {}'.format(n_vars_max))
print('Počet omezujících podmínek = {}'.format(n_constraints_max))
print('Doba řešení =  {} ms'.format(solver.wall_time()))
print()
print('Hodnota kriteriální funkce z = {:6.3f}'.format(solver.Objective().Value()))
for j in range(n_vars_max):
    print('{} = {:6.3f}'.format(x_max[j], x_max[j].solution_value()))
print()
h = 1/solver.Objective().Value()
```

```
print('Cena hry h (= 1/z) = {:6.3f}'.format(h))
for j in range(n_vars_max):
    print('alfa_' + str(j+1) + ' = {:6.3f}'.format(x_max[j].solution_value() * h))
```

```
Počet proměnných = 4
Počet omezujících podmínek = 5
Doba řešení =  74 ms

Hodnota kriteriální funkce z =  0.140
x1 =  0.000
x2 =  0.000
x3 =  0.080
x4 =  0.060

Cena hry h (= 1/z) =  7.143
alfa_1 =  0.000
alfa_2 =  0.000
alfa_3 =  0.571
alfa_4 =  0.429
```

2.8.4 Úloha minimalizace

```
lower_limits_min = [0, 0, 0, 0, 0] # min
x_min, n_vars_min = define_variables(solver, c_min, lower_limits_min)     # definice promě
n_constraints_min = set_constraints(solver, x_min, A_min, b_min, 'min')    # nastavení ome
set_objective(solver, x_min, c_min, 'min')            # nastavení cílové funkce

solver.Solve()
```

```
0
```

Výpis výsledků (minimalizace):

```
print('Počet proměnných = {}'.format(n_vars_min))
print('Počet omezujících podmínek = {}'.format(n_constraints_min))
print('Doba řešení =  {} ms'.format(solver.wall_time()))
print()
print('Hodnota kriteriální funkce z = {:6.4f}'.format(solver.Objective().Value()))
for j in range(n_vars_min):
    print('{} = {:6.4f}'.format(x_min[j], x_min[j].solution_value()))
print()
h = 1/solver.Objective().Value()
print('Cena hry h (= 1/z) = {:6.3f}'.format(h))
for j in range(n_vars_min):
    print('beta_' + str(j+1) + ' = {:6.3f}'.format(x_min[j].solution_value() * h))
```

```
Počet proměnných = 5
Počet omezujících podmínek = 4
Doba řešení =  123 ms

Hodnota kriteriální funkce z = 0.1400
x1 = 0.0400
x2 = 0.1000
```

```
x3 = 0.0000
x4 = 0.0000
x5 = 0.0000

Cena hry h (= 1/z) =  7.143
beta_1 =  0.286
beta_2 =  0.714
beta_3 =  0.000
beta_4 =  0.000
beta_5 =  0.000
```

3. Shrnutí získaných výsledků

Cena hry:

$z^* = 0.14 => h = \frac{1}{z^*} = 7.143$

Duální úloha:

$\alpha_1^* = x_1 * h = 0.04 * 7.143 = 0.286$
$\alpha_2^* = x_2 * h = 0.1 * 7.143 = 0.714$
$\alpha_3^* = x_3 * h = 0 * 7.143 = 0$
$\alpha_4^* = x_4 * h = 0 * 7.143 = 0$
$\alpha_5^* = x_5 * h = 0 * 7.143 = 0$

$=> \alpha^* = [0.286, 0.714, 0, 0, 0]^T$

Primární úloha:

$\beta_1^* = y_1 * h = 0 * 7.143 = 0$
$\beta_2^* = y_2 * h = 0 * 7.143 = 0$
$\beta_3^* = y_3 * h = 0.04 * 7.143 = 0.571$
$\beta_4^* = y_4 * h = 0.04 * 7.143 = 0.429$

$=> \beta^* = [0, 0, 0.571, 0.429]^T$

4. Závěr

Cena hry celkem činila 7.143.
Hráč 1 by měl volit strategii α_1 28.6 % času, strategii α_2 pak 71.4 % času.
Hráč 2 by měl volit strategii β_3 57.1 % času, strategii β_4 pak 42.9 % času.

0 s vyplněno 9:25