Będzin, 10.12.2020

MK group: 2
Section: C
Laboratory task: 3

Optimization and Decision Making Laboratory

Game Theory

Authors: Artur Oleksiński
Stefan Thomalla

Artur Oleksiński   Set 9 OaDM MAKRO gr 2   10.12.2020r
Stefan Thomalla

Zero-sum games:

Game 2×2

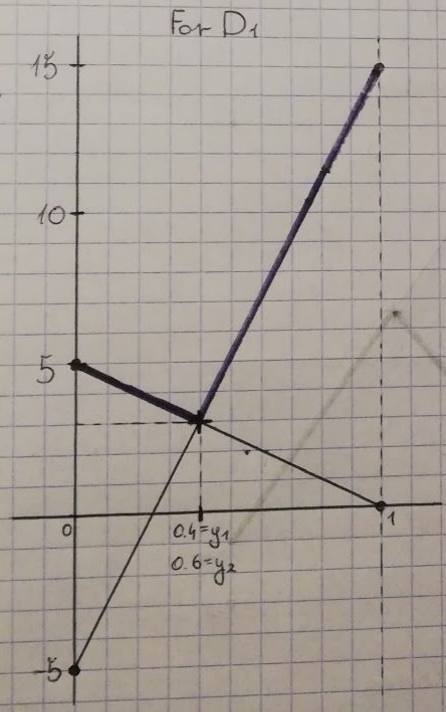
D_1 \ D_2	1	2	
1	5	-5	→ 5
2	0	15	→ 15
	↓	↓	
	0	-5	

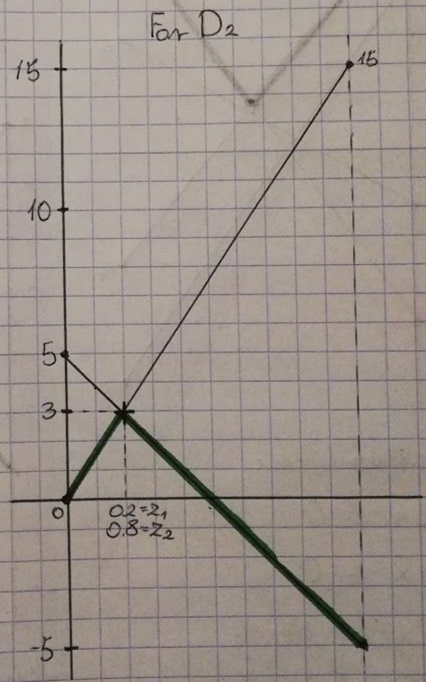
Safe strategy:

Decision:

$D_1(1)\ D_2(1)$

Mixed strategy:





$$J = [5 \cdot y_1] \cdot z_1 + [(-5) y_1 + 15 \cdot y_2] \cdot z_2 =$$

$$= (5 \cdot 0.4) \cdot 0.2 + [(-5) \cdot 0.4 + 15 \cdot 0.6] \cdot 0.8 =$$

$$= 0.4 + 5.6 = 6$$

Game $2 \times n$, $n=4$

p2

$D_1 \backslash D_2$	1	2	3	4	
1	-2	6	8	-6	→ 8
2	4	-8	-4	8	→ 8
	↓ -2	↓ -8	↓ -4	↓ -6	

Safe strategies:

Decisions:

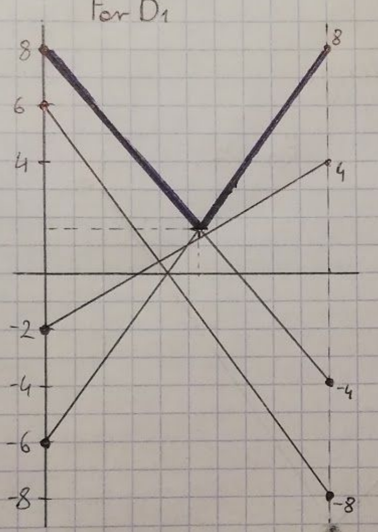
$D_1(1)\,D_2(1) = -2$

$D_1(2)\,D_2(1) = 4$

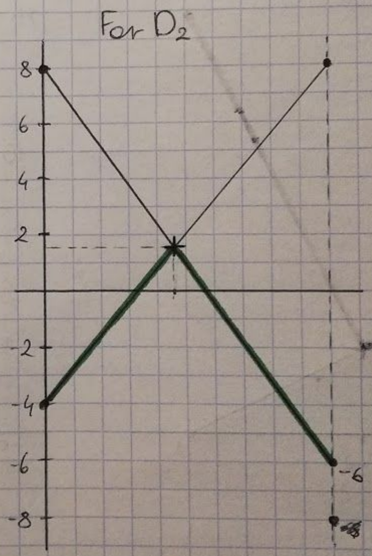
Mixed strategies:

$D_1 \backslash D_2$	3	4
1	8	-6
2	-4	8

For D_1



For D_2



Decision lines:

$$\begin{cases} -12x + 8 = 0 \\ 14x - 6 = 0 \end{cases}$$

$-12x + 8 = 14x - 6$

$-26x = -14 \quad // : (-26)$

$x = \frac{7}{13} \rightarrow y_1$

$1 - y_1 = y_2 = \frac{6}{13}$

Decision lines:

$$\begin{cases} -14x + 8 = 0 \\ 12x - 4 = 0 \end{cases}$$

$-14x + 8 = 12x - 4$

$26x = 12$

$x = \frac{6}{13} \rightarrow z_1$

$1 - z_1 = z_2 = \frac{7}{13}$

p3 $J = (8 \cdot y_1 - 4 \cdot y_2) \cdot z_1 + (-6 y_1 + 8 y_2) \cdot z_2 =$

$= \left(8 \cdot \frac{7}{13} - 4 \cdot \frac{6}{13}\right) \cdot \frac{6}{13} + \left(-6 \cdot \frac{7}{13} + 8 \cdot \frac{6}{13}\right) \cdot \frac{7}{13} =$

$= \frac{18}{13}$

Game n x m

D_1 \ D_2	1	2	3	4	5	
1	-5	3	2	-7	5	→ 5
2	3	-2	1	8	-6	→ 8
3	-2	0	-4	-4	-3	→ 0
4	4	-8	5	4	1	→ 5
5	-7	9	-7	-6	2	→ 9
	↓ -7	↓ -8	↓ -7	↓ -7	↓ -6	

Safe decision

Decision

$D_1(3)\, D_2(5) = -3$

LP:

$$
\begin{aligned}
-5y_1 + 3y_2 + 2y_3 - 7y_4 + 5y_5 &\leq J \\
3y_1 - 2y_2 + y_3 + 8y_4 - 6y_5 &\leq J \\
-2y_1 + 0 - 4y_3 - 4y_4 - 3y_5 &\leq J \\
4y_1 - 8y_2 + 5y_3 + 4y_4 + y_5 &\leq J \\
-7y_1 + 9y_2 - 7y_3 - 6y_4 + 2y_5 &\leq J \\
y_1 + y_2 + y_3 + y_4 + y_5 &= 1
\end{aligned}
$$

Non-zero sum games

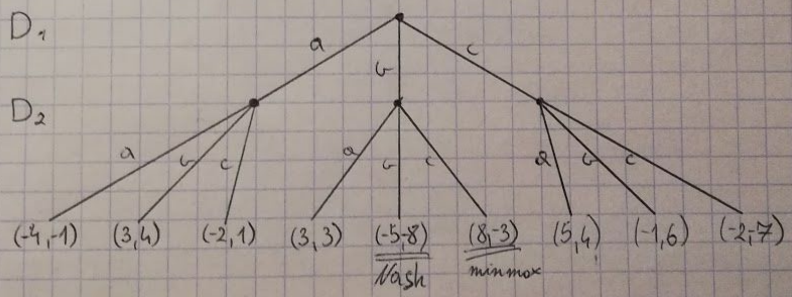
Nash and Minmax

D_1 \ D_2	a	b	c
a	-4	3	-2
b	3	-5	8
c	5	-1	-2
	↓ -4	↓ -5	↓ -2

D_1 \ D_2	a	b	c	
a	-1	4	1	→ -1
b	3	-8	-3	→ -8
c	4	6	-7	→ -7

Choices: (Nash equilibriums)

$$\left.\begin{aligned} D_1 D_2 &= (1,1) = (-4,-1) \\ D_1 D_2 &= (2,2) = (-5,-8) \\ D_1 D_2 &= (3,3) = (-2,-7) \end{aligned}\right\} \min = (2,2) = (-5,-8)$$



Minmax

D_1 \ D_2	a	b	c
a	-4	3	-2
b	3	-5	8
c	5	-1	-2
	↓ -4	↓ -5	↓ -2

D_1 \ D_2	a	b	c	
a	-1	4	1	→ 4
b	3	-8	-3	→ 3
c	4	6	-7	→ 6

Minmax strategy:

$D_1(2)\ D_2(3) = (8,-3)$

von Stackelberg

A

$D_1 \backslash D_2$	a	b	c
a	-4	3	-2
b	3	-5	8
c	5	-1	-2

B

$D_1 \backslash D_2$	a	b	c
a	-1	4	1
b	3	-8	-3
c	4	6	-7

$$\left.\begin{array}{l} R(a) = \{1\} \rightarrow (a,a) = -4 \\ R(b) = \{2\} \rightarrow (b,b) = -5 \\ R(c) = \{3\} \rightarrow (c,c) = -2 \end{array}\right\} \max \rightarrow (c,c) = \underline{\underline{-2, -7}}$$

Code presentation:

SafeStrategy.py output:

```
Payoff matrices:
 [[-2  6  8 -6]
 [ 4 -8 -4  8]]
Minimums from columns:
 [(1, -2), (2, -8), (3, -4), (4, -6)]
Safe Strategy choices for player1:
 [(1, -2)]
Maximums from rows:
 [(1, 8), (2, 8)]
Safe Strategy choices for player2:
 [(1, 8), (2, 8)]

Process finished with exit code 0
```

Nash.py output:

```
Pay-off matrix of player1:
 [[-4  3 -2]
 [ 3 -5  8]
 [ 5 -1 -2]]
Pay-off matrix of player2:
 [[-1  4  1]
 [ 3 -8 -3]
 [ 4  6 -7]]
Options of player1:
 [[0, 0, -4], [1, 1, -5], [2, 0, -2], [2, 2, -2]]
Options of player1:
 [[0, 0, -1], [1, 1, -8], [2, 2, -7]]
Choices:
 [([0, 0], -4, -1), ([1, 1], -5, -8), ([2, 2], -2, -7)]
```

```
Pay-off matrix of player1:
 [[-4  3 -2]
 [ 3 -5  8]
 [ 5 -1 -2]]
Pay-off matrix of player2:
 [[-1  4  1]
 [ 3 -8 -3]
 [ 4  6 -7]]
Options of player1:
 [[0, 0, -4], [1, 1, -5], [2, 0, -2], [2, 2, -2]]
Options of player1:
 [[0, 0, -1], [1, 1, -8], [2, 2, -7]]
Choices:
 [([0, 0], -4, -1), ([1, 1], -5, -8), ([2, 2], -2, -7)]
Best choice if we're searching for minimum:
 ([1, 1], -5, -8)
```

```
Pay-off matrix of player1:
 [[-4  3 -2]
 [ 3 -5  8]
 [ 5 -1 -2]]
Pay-off matrix of player2:
 [[-1  4  1]
 [ 3 -8 -3]
 [ 4  6 -7]]
Options of player1:
 [[0, 0, -4], [1, 1, -5], [2, 0, -2], [2, 2, -2]]
Options of player1:
 [[0, 0, -1], [1, 1, -8], [2, 2, -7]]
Choices:
 [([0, 0], -4, -1), ([1, 1], -5, -8), ([2, 2], -2, -7)]
Best choice if we're searching for max:
 ([0, 0], -4, -1)

Process finished with exit code 0
```

MinMax.py output:

```
 Pay-off matrix of player1:
 [[-4  3 -2]
 [ 3 -5  8]
 [ 5 -1 -2]]
Pay-off matrix of player2:
 [[-1  4  1]
 [ 3 -8 -3]
 [ 4  6 -7]]
Options of player1:
 [[0, -4], [1, -5], [2, -2]]
Options of player2:
 [[0, 4], [1, 3], [2, 6]]
Minmax strategy:
 [2, 1]
```

vonStuckelberg.py output:

```
Pay-off matrix of player1:
 [[-4  3 -2]
 [ 3 -5  8]
 [ 5 -1 -2]]
Pay-off matrix of player2:
 [[-1  4  1]
 [ 3 -8 -3]
 [ 4  6 -7]]
Rational responses:
        R0 = 0 = -1
        R1 = 1 = -8
        R2 = 2 = -7
Decision of the leader:
         [(2, 2), -2]
Coordinates:
         (2, 2)
Values:
        Player1:
                -2
        Player2:
                -7
```

Code repository:

https://github.com/ArturOle/GameTheory

If you want to run the programs, you should have installed Python 3.x interpreter and Numpy library
(in console write “pip install numpy” in case you haven’t)
Python codes, put all files in the same location which is the source folder. Utility classes are necessary!

Utility classes:
- table_provider.py
- exceptions.py

Task classes:
- Nash.py
- SafeStrategy.py
- MinMax.py
- vonStackelberg.py