INDIAN ACADEMY OF SCIENCE
SUMMER RESEARCH PROJECT

Code Development For Solving N-Player Strategic Game

Author:
Jayanth Prakash Kulkarni

Guide:
Prof.Y Narahari

November 8, 2015

Game Theory Laboratory
Dept. of Computer Science and Automation
Indian Institute of Science

Abstract

This project aims at solving a N-Player strategic game by finding the dominance, Nash equilibrium, maxmin and the minmax values and strategies.Thus is instrumental in establishing a clearer and deeper understanding of some basic concepts in game theory.

Acknowledgment

I would like to thank Prof.Y Narahari, my guide, for his invaluable time and guidance, for providing me a kind opportunity to work for this project, in such a serene environment as Indian Institute of Science. This project is an outcome of numerous interesting discussions that I had with him.I am deeply thankful to Indian Academy of Science for selecting me for this program.

Contents

1 Motivation

Game Theory finds its applications in various fields which include economics, political science, biology, computer science, logic and many more. Thus it is important to understand the fundamentals of such a diverse subject. This project provides a good start in understanding basics of game theory mainly through a computer science's perspective.

2 Definitions

Definition 1 (Strongly Dominant Strategy). *A strategy $s_i^* \epsilon S_i$ is said to be a strongly dominant strategy for player i if it strongly dominates every other strategy $s_i \epsilon S_i$. That is,* $\forall\ s_i \neq s_i^*$,

$$u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i})\ \forall s_{-i}\ \epsilon\ S_{-i}$$

Definition 2 (Weakly Dominant Strategy). *A strategy $s_i \epsilon S_i$ is said to be weakly dominant strategy for player i if it weakly dominates every other strategy $s_i \epsilon S_i$*

$$u_i(s_i^{'}, s_{-i}) \geq u_i(s_i, s_{-i})\ \forall s_{-i}\ \epsilon\ S_{-i}$$

And $u_i(s_i^{'}, s_{-i}) > u_i(s_i, s_{-i})$ For some $s_{-i} \epsilon S_{-i}$ The strategy s_i^* is said to be weakly dominate strategy s_i. Note that strict inequality is to be satisfied for at least one s_i.

Definition 3 (Very Weakly Dominant Strategy). *A strategy s_i^* is said to be a very weakly dominant strategy for player i if it very weakly dominates every other strategy $s_i \epsilon S_i$*

$$u_i(s_i^{'}, s_{-i}) \geq u_i(s_i, s_{-i})\ \forall s_{-i}\ \epsilon\ S_{-i}$$

Note that strict inequality need not be satisfied for any s_{-i} unlike in the case of weak dominance where strict inequality must be satisfied for atleast one s_i.

Definition 4 (Pure Strategy Nash Equilibrium). *Given a strategic form game* $\Gamma = \langle N, (S_i), (u_i) \rangle$ *a strategic profile* $(s_1^*, ..., s_n^*)$ *is called a pure strategy Nash equilibrium of* Γ *if*

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \; \forall s_{-i} \, \epsilon \, S_i, \, \forall i = 1, 2, ..., n$$

Definition 5 (Mixed Strategy Nash Equilibrium). *Given a strategic form game* $\Gamma = \langle N, (S_i), (u_i) \rangle$, *a mixed strategy profile (*$\sigma_1^*, ..., \sigma_n^*$*) is called a Nash equilibrium if* $\forall i \, \epsilon \, N$,

$$u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*) \quad \forall \sigma_i \, \epsilon \, \Delta(S_i).$$

Definition 6 (Maxmin Value and Maxmin Strategy). *Given a strategic form game,* $\Gamma = \langle N, (S_i), (u_i) \rangle$, *the maxmin value or security value of a player i (i = 1,...,n) is given by:*

$$\underline{v_i} = \max_{s_i \epsilon S_i} \; \min_{s_{-i} \epsilon S_{-i}} u_i(s_i, s_{-i})$$

Any strategy $s_i^* \epsilon S_i$ *that guarantees this payoff to player i is called a maxmin strategy or security strategy of the player i.*

Definition 7 (Minmax Value and Minmax Strategy). *Given a strategic form game,* $\Gamma = \langle N, (S_i), (u_i) \rangle$, *the minmax value of a player i (i = 1,....,n) is given by:*

$$\overline{v_i} = \min_{s_{-i} \epsilon S_{-i}} \; \max_{s_i \epsilon S_i} u_i(s_i, s_{-i})$$

Any strategy profile $s_{-i}^* \epsilon S_{-i}$ *if the other players that forces that payoff* $\overline{v_i}$ *on player i is called a minmax strategy profile (of the rest of the players) against player i.*

Maxmin, Minmax values and strategies for mixed strategies have similar definations as above, only that the strategies considered as not pure strategie but mixed strategies.

All definitions were referred from [1, 3]

3 Algorithms

In this section, detailed Pseudo Codes for the implemented algorithm are discussed. Gambit Software was used in order to generate the strategic game and the generated game was stored into a dynamic structure, organized according to the strategic profiles of each player. Later this systematic structure was used to calculate strongly, weakly and very weakly dominant strategies and their equilibrium, if they existed. In a further enhancement, the Nash equilibrium was also calculated, firstly the profiles were checked for pure strategy if not found mixed strategies were calculated. Then the minmax maxmin values and strategies were calculated using linear and quadratic programming paradigms.

The Input was taken in the form of a Gambit Format[3] and were solved using C programming language.

3.1 Strategy Traversal

Algorithm 1 Strategy Allocation

```
procedure STRATEGY(i, max)
    while i < max do
        if strategyset[i]+1 > action[i] then
            strategyset[i] = 1
            i++
        else
            strategyset[i]++
            return
        end if
    end while
end procedure
```

Algorithm 2 Strategy Allocation

```
procedure STRATEGYCOMPLEX(i, currentplayer)
    while i < numberofplayers do
        if strategyset[i]+1 > action[i] then
            strategyset[i] = 1
            if (i+1)<currentplayer then
                i++
                continue
            end if
            if i+1 == currentplayer then
                i=i+2
                continue
            end if
        else
            strategyset[i] = strategyset[i] + 1
            return
        end if
    end while
end procedure
```

Algorithm 3 Strategy Allocation

```
procedure CHANGESTRATEGYSET(currentplayer)
    if currentplayer == numberofplayers-1 then
        strategy(0,currentplayer)
    else if currentplayer==0 then
        strategy(1,numberofplayers)
    else
        strategyset[i] = strategyset[i] + 1
        return
    end if
end procedure
```

In the above mentiones algorithms, the first two functions were used to traverse through the strategies of players and the third function, a driver function was used to select one of the 2 functions based on the situation. This was done as the data was stored in the form of a strategy profile and in case of more than 2 player games this system helped in a systematic traversal.

3.2 Dominance

Here the algorithm used to find the strongly, weakly and very weakly dominant strategy and their equilibrium is discussed.

Algorithm 3 Dominance

```
procedure FINDDOMINANCE(i)
    for loop = 1;loop <= action[i];loop++ do
        initialize all strategyset to 1
        while 1 do
            strategyset[i] = currentstrategyloop
            curpayoff = findpayoff(strategyset,i)
            for j=loop-1;j>=1;j– do
                strategyset[i] = j
                cmppayoff = findpayoff(strategyset,i)
                if curpayoff<cmppayoff then
                    canbreak = 1
                    break
                end if
                if curpayoff == cmppayoff then
                    strong = 0
                end if
                if curpayoff>cmppayoff then
                    c1++
                end if
            end for
            if canbreak==1 then
                break from the while loop ;
            end if
            strategyset[i] ← currentstrategyloop
            for j=currentstrategyloop + 1;j¡=action[i];j++ do
                strategyset[i] ← j

```

```
            end for
            for k=0;k<numberofplayers;k++ do
                if k==i then
                    continue
                end if
                if strategyset[k]!=action[k] then
                    canprint = -1
                    break
                end if
                if canprint == 1 then
                    if strong == 1 then
                        strongdominanceequilibrium ← currentstrategy
                    end if
                    if c3==1 then
                        weakdominanceequilibrium ← currentstrategy
                    end if
                    veryweakdominanceequilibrium ← currentstrategy
                else
                    changestrategyset(i)
                    continue
                end if
            end for
        end while
    end for
end procedure
```

In this algorithm, the payoffs which were organized in the form of strategic profiles, were systematically traversed. The determination of the dominance and the dominance equilibrium of a particular strategy was achieved by performing a series of comparisons this can be observed quiet intuitively in the algorithm.

3.3 Pure Nash Equilibrium

As discussed in the previous chapter, Pure strategy Nash equilibrium is that equilibrium in which there is no benefit for any player to unilaterally deviate from his equilibrium strategy. The following algorithm shows how this equilibrium can be calculated.

Algorithm 4 Pure Strategy Nash Equilibrium

```
procedure PSNE
    for j=0;j<numberofstrategyprofiles;j++ do
        strategyset ← strategy.
        for i=0;i<numberofplayers;i++ do
            get the current payoff.
            for set[i]=set[i]-1;set ≥ 1; set[i]– do
                get comparing payoff
                if curpayoff<cmppayoff then
                    Break i^th loop
                end if
            end for
            for set[i]=set[i]+1;set ≤ action[i]; set[i]++ do
                get comparing payoff
                if curpayoff<cmppayoff then
                    Break i^th loop
                end if
            end for
        end for
        for n=0;n<numberofplayers;n++ do
            ispure ← 1
            Nashequilibrium[n] ← changestrategyset[n]
        end for
        if ispure ≠ 1 then
            call MSNE
        end if
    end for
end procedure
```

Here when there is no Pure Strategy Nash Equilibrium, the algorithm calls for MSNE function which calculates the Mixed Strategy Nash Equilibrium.

3.4 Mixed Strategy Nash Equilibrium

Given any Strategic game there has to be a Nash Equilibrium for that game. It is noted that if a game doesn't contain a PSNE, then that game must have a MSNE.In order to find the MSNE of a game a generic algorithm is followed.

Algorithm 5 Mixed Strategy Nash Equilibrium

```
procedure MSNE
    initialize ← mixedpayoff
    for i=0;i<numberofplayers;i++ do
        for loop=1;loop ≤ action[i];loop++ do
            Initialize strategyset ← 1
            Find curpayoff
            while 1 do
                strategyset[i] ← loop
                Find cmppayoff
                if i = 0 then
                    payoff[i][strategy[0]-1][strategy[1]-1]= cmppayoff
                else
                    payoff[i][strategy[1]-1][strategy[0]-1]= cmppayoff
                end if
                for k=0;k<numberofplayers;k++ do
                    if k==i then
                        continue;
                    end if
                    if strategyset[k]!=action[k] then
                        canprint = -1;
                        break;
                    end if
                    if canprint == 1 then
                        break
                    else
                        changestrategyset(i)
                        continue
                    end if
                end for
            end while
            initialize Powerset ⇐ Powersetcalculator
            Eleminate the pure strategy powersets and supports
            Consider only mixed strategy supports
```

```
            Initialize solvegauss ← supports
            Call gausseliminationsolver(solvegauss)
            if solution = negative then
                break
            else if if not considered payoff > considered then
                break
            else
                declare it to be mixed strategy
            end if
        end for
    end for
end procedure
```

3.5 Maxmin value and Strategy for Pure strategies

Algorithm 6 Maxmin value and Strategy for Pure strategies

```
procedure DRIVER FUNCTION
    for i=0;i¡numberofplayers;i++ do
        call maxminps(i)
        m ← maxminarray[i].index[1]
        for j=2;j≤action[i];j++ do
            if maxminarray[i].value[j]≥m then
                if maxminarray[i].value[j]=m then
                    store j
                    initialize check = 1
                else if check=1 then
                    for c=0;c≤action[i];c++ do
                        discard stored value c
                    end for
                    m = maxminarray[i].value[j]
                    sotre the index
                end if
            end if
        end for
    end for
end procedure
```

Algorithm 7 Maxmin value and Strategy for Pure strategies

```
procedure MAXMINPS(i)
    for loop=1;loop ≤ action[i];loop++ do
        Initialize strategyset ← 1
        Find curpayoff
        while 1 do
            strategyset[i] ← loop
            Find cmppayoff
            if curpayoff > cmppayoff then
                curpayoff ← cmppayoff
                curpayoff = cmppayoff
                store payoff and strategy
            end if
            for k=0;k<numberofplayers;k++ do
                if k==i then
                    continue;
                end if
                if strategyset[k]!=action[k] then
                    canprint = -1;
                    break;
                end if
                if canprint == 1 then
                    break
                else
                    changestrategyset(i)
                    continue
                end if
            end for
        end while
    end for
end procedure
```

The maxminps algorithm calculates the minimum value of all the payoffs and is stored. In the First algorithm 1 i.e. the driver procedure calculates the max over these set of min values.

3.6 Minmax value and Strategy for Pure strategies

Algorithm 8 Minmax value and Strategy for Pure strategies

```
procedure MINMAXPS(i)
    for loop=1;loop ≤ action[i];loop++ do
        Initialize strategyset ← 1
        Find curpayoff
        while 1 do
            strategyset[i] ← loop
            for player=0;player¡numberofplayers;player++ do
                if player=i then
                    continue
                end if
                Find cmppayoff
                if curpayoff < cmppayoff then
                    curpayoff ← cmppayoff
                    curpayoff = cmppayoff
                    store payoff and strategy
                end if
            end for
            for k=0;k<numberofplayers;k++ do
                if k==i then
                    continue;
                end if
                if strategyset[k]!=action[k] then
                    canprint = -1
                    break;
                end if
                if canprint == 1 then
                    break
                else
                    changestrategyset(i)
                    continue
                end if
            end for
        end while
    end for
end procedure
```

Algorithm 9 Minmax value and Strategy for Pure strategies

```
procedure Driver function
    for i=0;i¡numberofplayers;i++ do
        call minmaxps(i)
        for c=0;c<numberofplayers;c++ do
            if c=1 then
                continue
            end if
        end for
        for c=0;c<numberofplayers;c++ do
            if c==i then
                continue
            end if
            m ← minmaxarray[i].value[c][1]
            indexarray ← = minmaxarray[i].index[c][1]
            numberofvalues = 1
            for j=2;j≤action[c];j++ do
                if m≥ minmaxarray[i].value[c][j] then
                    if m = minmaxarray[i].value[c][j] then
                        check = 1;
                        increment numberofvalues
                        indexarray ← minmaxarray[i].index[c][j]
                    else
                        numberofvalues = 1
                        if check = 1 then
                            for k=1;k≤action[i];k++ do
                                indexarray ← 0
                            end for
                        end if
                        indexarray[c][numberofvalues] = minmaxarray[i].index[c][j];
                        m = minmaxarray[i].value[c][j]
                    end if
                end if
            end for
        end for
    end for
end procedure
```

As the minmax values and strategies are against a certain player, the computation is slightly more time consuming than the maxmin values and strategies. In the above algorithm the max values of all the players are stored and the min value over these max values would give the minmax value and strategy that a player plays against the considered player.

3.7 Maxmin and Minmax values over Mixed Strategies

The maxmin value of a player is the highest payoff the player can gaureentee himself even in the worst case when the other players are free to play any mixed strategies. The minmax values in the mixed strategies of a plauer i is the lowest payoff that the other players will be able to force on the player i when they choose mixed strategies that hurt player i the most.[ref]

In order to solve this problem the number of players are restricted to 2 for simplicity.The minmaximization problem here can be converted to linear programs. The following 2 linear programs describe the optimization problems for the 2 players considered.

Let $x = (x_1,...,x_m)$ and $y = (y_1, ...y_n)$ be the mixed strategies of player1 and player 2 respectively. p_{ij} is the payoff of player 1.

the expected payoff for player 1 according to probability theory definition of expected value is

Expected payoff for player 1

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_iy_j = v$$

In order to find a desirable solution for a linear programming program 2 difficulties are faced

- v is unknown
- The linear programming problem has no objective function.

These problems can be solved by replacing the unknown constant v by the variable x_{m+1} and then maximizing x_{m+1}, so that x_{m+1} will be equal to v.

Player 1 would find his optimal mixed strategy by using the simplex method to solve the linear programming problem:

$$
\begin{array}{c}
\text{Maximize} \quad x_{m+1}, \\
\text{subject to} \\
p_{11}\ x_1 + p_{21}\ x_2 + \ldots + p_{m1} x_m - x_{m+1} \geq 0 \\
p_{12}\ x_1 + p_{22}\ x_2 + \ldots + p_{m2} x_m - x_{m+1} \geq 0 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
p_{1n}\ x_1 + p_{2n}\ x_2 + \ldots + p_{mn} x_m - x_{m+1} \geq 0 \\
x_1 + x_2 + \ldots + x_m = 1
\end{array}
$$

and

$x_i \geq 0$, for $i = 1,2,\ldots,m$.

Similarly player 2 would conclude that his optimal mixed strategy is given by an optimal solution to the linear programming problem :

$$
\begin{array}{c}
\text{Maximize} \quad y_{m+1}, \\
\text{subject to} \\
p_{11}\ y_1 + p_{21}\ y_2 + \ldots + p_{m1} y_m - y_{m+1} \geq 0 \\
p_{12}\ y_1 + p_{22}\ y_2 + \ldots + p_{m2} y_m - y_{m+1} \geq 0 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
p_{1n}\ y_1 + p_{2n}\ y_2 + \ldots + p_{mn} y_m - y_{m+1} \geq 1 \\
y_1 + y_2 + \ldots + y_m = 1
\end{array}
$$

and

$y_i \geq 0$, for $j = 1,2,\ldots,m$.

These equations were referred from [2]

3.8 Quadratic Programming for solving 2 player non-zerosum games

The necessary and sufficient condition that a point be a Nash equilibrium of a two-player, nonzero-sum game with finite number of pure strategies is that the point be a solution of a single programming problem with linear constraints and a quadratic objective function that has a global maximum of zero. Every equilibrium point is a solution of this programming problem. For the case of a zero-sum game, the quadratic programming problem degenerates to the well-known dual linear programs associated with the game as shown in the earlier subsection.

A majority of this section is derived from [4]. Consider a two-player general sum game $\mathcal{G} = (\mathbf{P}, \Sigma, \mathbf{A}, \mathbf{B})$ with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$. Let $\mathbf{1}_m \in \mathbb{R}^{m\times 1}$ be the vector of all ones with m elements and let $\mathbf{1}_n \in \mathbb{R}^{n\times 1}$ be the vector of all ones with n elements.There is at least one Nash equilibrium $(\mathbf{x}^*, \mathbf{y}^*)$. If either Player were to play his/her Nash equilibrium, then the optimization problems for the players would be:

$$P_1 \begin{cases} \max & \mathbf{x}^T\mathbf{A}\mathbf{y}^* \\ s.t. & \mathbf{1}_m^T\mathbf{x} = 1 \\ & \mathbf{x} \geq \mathbf{0} \end{cases}$$

$$P_2 \begin{cases} \max & \mathbf{x}^{*T}\mathbf{B}\mathbf{y} \\ s.t. & \mathbf{1}_n^T\mathbf{y} = 1 \\ & \mathbf{y} \geq \mathbf{0} \end{cases}$$

Individually, these are linear programs. The problem is, we don't know the values of $(\mathbf{x}^*, \mathbf{y}^*)$ *a priori*. However, we can draw insight from these problems.

We can find a third Nash equilibrium for the Chicken game using this approach. Recall we have:

$$\mathbf{A} = \begin{bmatrix} 0 & -1 \\ 1 & -10 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \\ -1 & -10 \end{bmatrix}$$

Then our quadratic program is:

$$
(1)\quad \begin{cases}
\max \quad \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -20 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \alpha - \beta \\
s.t. \quad \begin{bmatrix} 0 & -1 \\ 1 & -10 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & -10 \end{bmatrix} - \begin{bmatrix} \beta & \beta \end{bmatrix} \leq \begin{bmatrix} 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1 \\
\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 1 \\
\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{cases}
$$

This simplifies to the quadratic programming problem:

$$
(2)\quad \begin{cases}
\max \quad -20x_2y_2 - \alpha - \beta \\
s.t. \quad -y_2 - \alpha \leq 0 \\
y_1 - 10y_2 - \alpha \leq 0 \\
-x_2 - \beta \leq 0 \\
x_1 - 10x_2 - \beta \leq 0 \\
x_1 + x_2 = 1 \\
y_1 + y_2 = 1 \\
x_1, x_2, y_1, y_2 \geq 0
\end{cases}
$$

An optimal solution to this problem is $x_1 = 0.9$, $x_2 = 0.1$, $y_1 = 0.9$, $y_2 = 0.1$. This is a third Nash equilibrium in mixed strategies for this instance of Chicken. Identifying this third Nash equilibrium in Matlab is shown in Figure below. In order to correctly input this problem into Matlab, we need to first write the problem as a proper quadratic program. This is done by letting the vector of decision variables be:

$$
\mathbf{z} = \begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix}
$$

Then the quadratic programming problem for Chicken is written as:

$$
(3)\quad \begin{cases}
\max\ \dfrac{1}{2}\begin{bmatrix} x_1 & x_2 & y_1 & y_2 & \alpha & \beta \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -40 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix} \\
s.t. \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & -10 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 \\ 1 & -10 & 0 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \alpha \\ \beta \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\infty \\ -\infty \end{bmatrix}
\end{cases}
$$

Note, before you enter this into Matlab, you must transform the problem to a minimization problem by multiplying the objective function matrices by -1.

```
>> Q = [[0 0 0 0 0 0];[0 0 0 40 0 0];[0 0 0 0 0 0];[0 0 0 0 0 0];[0 0 0 0 0 0];[0 0 0 0 0 0]]

Q =

     0     0     0     0     0     0
     0     0     0    40     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0

>> c = [0;0;0;0;1;1];
>> A = [[0 0 0 -1 -1 0];[0 0 1 -10 -1 0];[0 -1 0 0 0 -1];[1 -10 0 0 0 -1]];
>> b = [0;0;0;0];
>> H = [[1 1 0 0 0 0 ];[0 0 1 1 0 0]]

H =

     1     1     0     0     0     0
     0     0     1     1     0     0

>> r = [1;1];
>> l = [0;0;0;0;-inf;-inf];
>> u = [inf;inf;inf;inf;inf;inf];
>> [x obj] = quadprog(Q,c,A,b,H,r,l,u);
Warning: Your Hessian is not symmetric. Resetting H=(H+H')/2.
> In quadprog at 260
Warning: Large-scale algorithm does not currently solve this problem formulation,
using medium-scale algorithm instead.
> In quadprog at 291
Optimization terminated.
>> x

x =

    0.9000
    0.1000
    0.9000
    0.1000
   -0.1000
   -0.1000

>>
>> |
```

This figure is a screenshot of the matlab implementation of the problem.

4 Result and Discussion

4.1 Observations

The following observations observations and relation among the concepts could be established, when the topics discussed in the 3^{rd} section were implemented.[1]

- Dominant strategies may or may not exist. A strictly dominant strategy if exists will be unique.The same does not hold good for weakly and very weakly dominant strategy.
- A Strictly dominant strategy equilibrium if exists, a weakly and very weakly dominant strategy equilibrium exits for the same strategy profile.
- Any dominant strategy equilibrium is a pure strategy Nash equilibrium.
- Nash equilibrium may not always exist.
- Multiple Nash equilibria can also exist. Battle of Sexes game illustrates this.
- Sum of the utilities obtained in a Nash equilibrium may not be maximal.Pigou's Network game is one such example.
- Nash equilibrium provides insurance for a player against his own unilateral deviations only .May not provide insurance against unilateral deviations by other players.Does not provide insurance against multilateral deviations.
- Nash equilibrium may not correspond to a socially optimal outcome.
- Nash equilibrium is a much weaker notion of equilibrium than a dominant strategy equilibrium, because Nash equilibrium is the best response against Nash equilibrium strategies of the rest of the players whereas dominant strategy equilibrium offers a best response irrespective of the strategies of rest of the players.
- Nash equilibrium need not be a dominant strategy equilibrium.

- Maxmin strategy of a player is the best possible payoff that can be guaranteed even in the worst case when the other players are free to choose any strategy. Hence otherwise known as *security* or *no-regret* strategy.
- There can exist multiple maxmin strategies.
- Payoff of a player in Nash equilibrium profile is at least the maxmin value of the player.In general the maxmin strategy differ from Nash equilibrium profiles.Thus different from dominant strategies also.
- Minmax value of a player is the lowest payoff that can be forced on him,when the other players choose strategies that hurt that player the most.
- Nash equilibrium is always greater than or equal to minmax value which in tern is greater than or equal to maxmin value.

4.2 Results

All the above mentioned algorithms were solved using C programming language except for quadratic programming problem which was solved in Matlab using optimization toolbox [5]. Several test cases were taken into consideration and the results were verified with already established results. Thus proving the correctness of the implementation.

The complete implementation of the code will be put up on the following GitHub link.

`https://github.com/Jayanth-Kulkarni`

References

[1] Game Theory and mechanism Design Authored by **Y.Narahari**
`http://lcm.csa.iisc.ernet.in/hari/book.html`

[2] Introduction to operations research Authored by **Hillier / Lieberman**
`http://www.mhhe.com/engcs/industrial/hillier/`

[3] Gambit Software
`http://gambit.sourceforge.net/`

[4] Two-Person Nonzero-Sum Games and Quadratic Programming Authored by
O. L. MANGASARIAN AND H. STONE
`http://www.sciencedirect.com/science/article/pii/0022247X64900216`

[5] MATLAB and Statistics Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.