List of Tables

List of Figures

Contents

I. Introduction

1.1 Objectives

The game theory project consists of three problems and in all problems, I will determine:

- If the game is zero-sum or not
- Pure Nash equilibria
- The dominated strategies, and if there are the dominant strategies

1.2 Project structures

The project structure will be divided into three parts following the requirement and also an extra part. The first section is a game with two players and they have two strategies, the second part will be two players but multiple strategies per player. The next part is to make a three players game with three actions for each, and the extra part I will work with multiple player with multiple strategies for each player.

II. Programs and Materials

For the graphical user interface part, I create a web application by using `React` for front-end part and `Golang` for backend server. The reason I chose this stack because it is easier to create UI in web base and also I am familiar with Javascript and Golang. More important, Golang is much faster than Python (I did a version in Python but it is slower than Golang).

Requirement

- NodeJS
- Go

Dependencies

- React
- Gin and Static middleware for Golang

Installation, Running and testing

- Download and extract the project
- `npm install` (install dependencies)
- `npm run-script build`
- `go run ./main.go`
- Use your browser and go to `localhost:8082`

Methodology

Problem 1 - game of 2 players with 2 strategies

In this problem, There will be two player named as A and B. The strategy that a player can input utilizing number begun from 0, such as strategy ”1”, strategy ”2”,..., and it is interpreted as the reward tables below.

Table 1: (Reward for Player A)

		Player B	
	Action	0	1
Player A	0	2	1
	1	0	1

Table 2: (Reward for Player B)

		Player A	
	Action	0	1
Player B	0	1	0
	1	0	2

User interface:

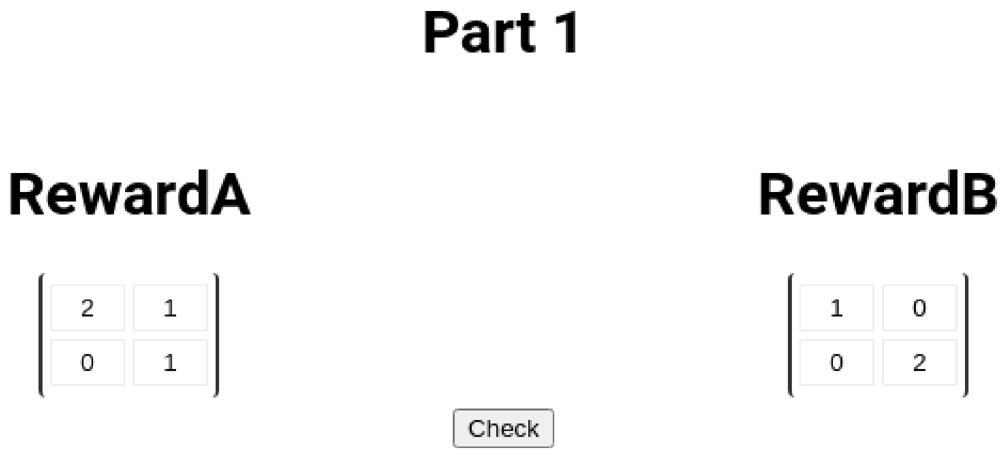

Figure 1: Problem 1 Graphical Interface

We created the interface same as our input so that the strategy and reward will be represented as array `[[2 1] [0 1]]` for player A and `[[1 0] [0 2]]` for player B.

Zero sum

In game theory, a zero-sum game is a mathematical representation of a situation in which an advantage that is won by one of two sides is lost by the other. In order to find out, we will check the reward of both players (the total gains and the total losses), if the reward sum equal to zero (the total gains are ended up and total lesses are subtracted).

```go
func CompareMatrix(mat1 [][]int, mat2 [][]int) bool {
    row := len(mat1)
    if row != len(mat2) {
        return false
    }
    for i := 0; i < row; i++ {
        col := len(mat1[i])
        if col != len(mat2[i]) {
            return false
        }
        for j := 0; j < col; j++ {
            if mat1[i][j] != mat2[i][j] {
                return false
            }
        }
    }
    return true
}
```

Pure Nash equilibrium state

In pure strategy, if player A play a (with probability 1), player B can play for example the same action a but with probability 1 so that there will be no random play (pure nash equil state). A Nash equilibrium is a set of strategies, one for each player, such that no player has incentive to change his or her strategy given what the other players are doing. The first problem is two players versus game so that we can use brute force method to check every strategy, in each strategy, we find if there exist a better action for a player.

```go
func CheckBetter(pA [][]int, pB [][]int, i int, j int) bool {
    //pA check
    for k := 0; k < 2; k++ {
        if pA[k][j] > pA[i][j] {
            return true
        }
    }
    //pB check
    for k := 0; k < 2; k++ {
        if pB[i][k] > pB[i][j] {
            return true
        }
    }
    return false
}

result = result + "Nash equilibrium state <br/>player A, player B
   :<br/>"
for i := 0; i < 2; i++ {
    for j := 0; j < 2; j++ {
        existBetter := CheckBetter(pA, pB, i,j)
        if !existBetter {
            fmt.Printf("%d,%d\n", i, j)
            result = result + fmt.Sprintf("%d,%d<br/>", i, j)
        }
    }
}
```

Mixed Strategy

A mixed strategy of player i is a measure of probability pi defined on the set of pure strategies of player i. We denote the set Pi the set of mixed strategies of player i. pi, si is the probability that i will play the pure strategy si. pi in Pi therefore corresponds to a mixed strategy of player i.