Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика

Теория игр

Отчет по лабораторной работе №2

Работу выполнили:
Обиджанов Алишер
Казаков Андрей
Кузнецов Павел **Преподаватель:**

Свинцов М.В.

Санкт-Петербург
2023

1 Постановка задачи

1. Реализуйте возможность ввода данных из файла в формате JSON, который содержит матрицу игры.
2. Упростите платежную матрицу путем анализа доминирующих стратегий.
3. Если это возможно найдите решение игры в чистых стратегиях. Определите оптимальные стратегии и соответствующую цену игры.
4. Если решение в чистых стратегиях найти невозможно, примените симплекс- метод для поиска седловой точки в смешанных стратегиях. Определите сме- шанные стратегии и соответствующую цену игры

2 Реализация

2.1 Реализуйте возможность ввода данных из файла в формате JSON, который содержит матрицу игры.

Пример json строки:

```
{"matrix":[[10,4,11,7],[7,6,8,20],[6,2,1,11]]}
```

Парсинг строки:

```
def parse_problem(json_str):
    try:
        data = json.loads(json_str)

        if "matrix" in data:
            matrix = data["matrix"]

            if isinstance(matrix, list) and all(isinstance(line, list) for line in matrix):
                return np.array(matrix)
            else:
                raise ValueError("Неверный формат матрицы в JSON.")
        else:
            raise ValueError("Отсутствует поле 'matrix' в JSON.")

    except json.JSONDecodeError as e:
        raise ValueError(f"Ошибка декодирования JSON: {e}")
```

2.2 Упрастите платежную матрицу путем анализа доминирующих стратегий

Сперва разделим каждый элемент на максимальный общий делитель и избавимся от отрицательных значений

```
def simplefy_matrix(matrix):
min_modul_of_number = 1e9
min_number = 1e9

for line in matrix:
    for num in line:
        if abs(num) < min_modul_of_number and num != 0:
            min_modul_of_number = abs(num)
        if num < min_number:
            min_number = num

findDel = True

for line in matrix:
    if not findDel:
        break
    for num in line:
        if num % min_modul_of_number != 0:
            findDel = False
            break

if findDel:
    min_number /= min_modul_of_number
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            matrix[i][j] = matrix[i][j] / min_modul_of_number

if min_number < 0:
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            matrix[i][j] += -min_number
return matrix
```

Далее будем сокращать строки и столбцы путем анализа доминирующих стратегий до тех пор пока это возможно

```
def reduce_matrix(matrix):
abbreviated = True
deleteLines = []
deleteColumn = []

while abbreviated:
    ilines = list(set(range(matrix.shape[0]))-set(deleteLines))
    icolumns = list(set(range(matrix.shape[1]))-set(deleteColumn))

    abbreviated = False
    for iline1 in ilines:
        for iline2 in ilines:
            is_reduce = True
            if iline1 == iline2:
                continue
            for column in icolumns:
                if not matrix[iline1][column] >= matrix[iline2][column]:
                    is_reduce = False
                    break
            if not is_reduce:
                continue
            else:
                deleteLines.append(iline2)
                abbreviated = True

    for icolumn1 in icolumns:
        for icolumn2 in icolumns:
            is_reduce = True
            if icolumn1 == icolumn2:
                continue
            for iline in ilines:
                if not matrix[iline][icolumn1] >= matrix[iline][icolumn2]:
                    is_reduce = False
                    break
            if not is_reduce:
                continue
            else:
                deleteColumn.append(icolumn1)
                abbreviated = True

matrix = np.delete(matrix, deleteColumn, axis=1)
matrix = np.delete(matrix, deleteLines, axis=0)
return matrix, sorted(deleteLines), sorted(deleteColumn)
```

2.3 Если это возможно найдите решение игры в чистых стратегиях. Определите оптимальные стратегии и соответствующую цену игры.

В чистых стратегиях можно решить задачу только в том случае, если у матрицы есть седловая точка. Ее можно найти путем вычисления минимаксных значений. Так же её можно найти, если мы сократили нашу изначальную матрицу до размера 1:1. Использую второй способ.

```
def strategies(matrix, deleteLines, deleteColumn):
Pa = []
Qb = []

if matrix.shape[0] == 1 and matrix.shape[1] == 1:
    for i in range(len(deleteLines) + 1):
        Pa.append(0) if i in deleteLines else Pa.append(1)
    for i in range(len(deleteColumn) + 1):
        Qb.append(0) if i in deleteColumn else Qb.append(1)
return Pa, Qb
```