# ANALYSIS OF QUEUEING SYSTEMS USING GAME THEORY

A Project Report Submitted

for the Course

**MA498 Project I**

*by*

**Karan Gupta**

(Roll No. 180123064)

**Ashish Kumar Barnawal**

(Roll No. 180123006)

iitglogo-eps-converted-to.pdf

*to the*

**DEPARTMENT OF MATHEMATICS**

**INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**

**GUWAHATI - 781039, INDIA**

*November 2021*

# CERTIFICATE

This is to certify that the work contained in this project report entitled "Analysis of Queueing Systems using Game Theory" submitted by Karan Gupta (Roll No. 180123064 and Ashish Kumar Barnawal (Roll No.: 180123006) to the Department of Mathematics, Indian Institute of Technology Guwahati towards partial requirement of Bachelor of Technology in Mathematics and Computing has been carried out by him/her under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

OR

It is also certified that, along with literature survey, a few new results are established/computational implementations have been carried out/simulation studies have been carried out/empirical analysis has been done by the student under the project.

Turnitin Similarity: 25 %

| | |
|---|---|
| Guwahati - 781 039 | (Prof. N. Selvaraju) |
| November 2021 | Project Supervisor |

# ABSTRACT

In this project, we have performed a survey of the existing literature on the applications of game theory in order to optimize the routing strategy across a network of queues with 2 or more players. Each player has a set of routes from which they have to choose a routing strategy such that they can minimize the mean sojourn time of their customers. We have explored weighted congestion games and seen how they relate to queueing games. We provide an algorithm to compute the payoff matrix (with payoffs being mean sojourn time of customers) from a given model of a queueing network game.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this paper, our main goal is to analyze the application of game theory to networks of queues in order to optimize a payoff function associated with the queueing game. We explore non-cooperative queueing games in Chapter 2 by presenting a model and stating theorems that prove/disprove the existence of Nash Equilibrium for certain sub-classes of queueing games. In Chapter 3, we have introduced an algorithmic approach to check the existence of pure strategy (mixed also?) Nash Equilibria for non-cooperative N-player games on a generalized network of queues, with a predefined strategy space for each player. We have also explored and analyzed the Best-response algorithm which shows that, for a game with a continuous strategy space, a pure-strategy Nash Equilibrium always exists. In Chapter 4, we conclude the paper with a brief overview of our findings; and provide an array of avenues which would look to explore and work on, in the future.

## 1.1 Queueing Theory

**Definition 1.1.1.** $M/M/1$ Queue. An M/M/1 queue is a single-server queue, and according to the Kendall's notation, has arrival rate($\lambda$) following the Markovian($M$) distribution, which means the inter-arrival times of customers entering the queue are exponential. The service rate($\mu$) of the queue is also Markovian($M$) and hence is an exponential service time. The maximum number of customers in the queue at the same time is unbounded or infinite.

**Theorem 1.1.2.** *The expected waiting time for an M/M/1 queue with arrival rate $\lambda$ and service rate $\mu$ is equal to $\frac{1}{\mu-\lambda}$.*

*Proof.* Let n be the number of customers at a given time, in the queue. We can make the flow-balance equations for n $\geq$ 1 and for the state with no customers as follows

$$(\lambda+\mu)p_n = \mu p_{n+1} + \lambda p_{n-1} \tag{1.1}$$

$$\lambda p_0 = \mu p_1 \tag{1.2}$$

where $p_i$ is the long-term fraction of time with $i$ customers in the system. Chapter 3 of [**?**] gives an explanation for using the set of long-term fractions $p_n$ for making flow balance equations and prove the above results. The following figure shows a state diagram for the number of customers in the system at a time along with the rates of transition to the next or previous state.

To solve above equations for $\{p_n\}$, we can use the **Method of Operators**.

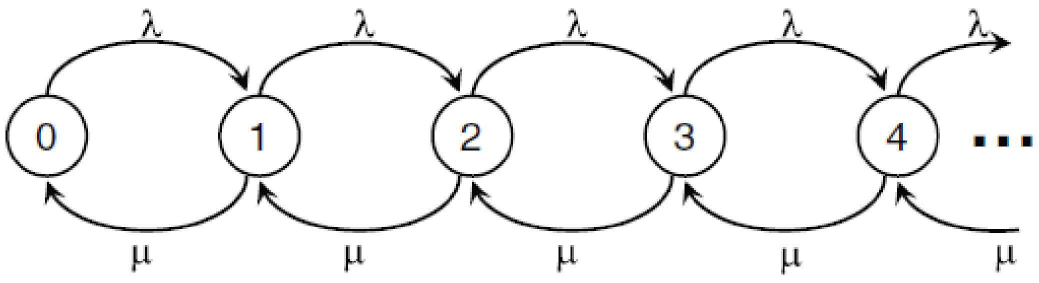

Figure 1.1: Rate transition diagram for M/M/1 queue

The equations **??** and **??** can be rewritten as

$$p_{n+1} = \frac{\lambda+\mu}{\mu}p_n - \frac{\lambda}{\mu}p_{n-1} \tag{1.3}$$

$$p_1 = \frac{\lambda}{\mu}p_0 \tag{1.4}$$

Let $\rho = \lambda/\mu$, then **??** becomes

$$p_{n+1} = (\rho+1)p_n - \rho p_{n-1} \tag{1.5}$$

We define a linear operator $D$ on $\{p_n\}$ such that

$$Dp_n = p_{n+1} \qquad \text{and in turn,}$$
$$D^m p_n = p_{n+m}$$

Writing **??** with $n = n + 1$ and putting every term on LHS, we get

$$\mu p_{n+2} - (\lambda+\mu)p_{n+1} + \lambda p_n = 0 \qquad \text{(n} \geq 0)$$

Converting to linear operator form, we have

$$[\mu D^2 - (\lambda+\mu)D + \lambda]p_n = 0 \tag{1.6}$$

subj. to boundary conditions **??** and $\sum_{n=0}^{\infty} p_n = 1$ Solving **??** gives $\{p_n\}$ as the solution. On factorizing **??**, we get

$$(D-1)(\mu D - \lambda)p_n = 0,$$

and so the final form of $p_n$ is $p_n = k_1(1)^n + k_2\rho^n = k_1 + k_2\rho^n$

The stability condition for queues requires $\rho < 1$ and as $\sum_{n=0}^{\infty} p_n = 1$, $k_1$ has to be 0, otherwise the summation will go to infinity. Also, from **??** and $p_1 = k_2\rho$, we have $k_2 = p_0$. Hence, $p_n = p_0\rho^n$. Summing $\{p_n\}$ over all $n$ gives $p_0 = 1 - \rho$.

Now, we find the expected number of customers in the queue in steady state, Let L denote the value, then

$$L = \sum_{n=0}^{\infty} np_n = (1-\rho)\sum_{n=0}^{\infty} n\rho^n$$

As $\sum_{n=1}^{\infty} \rho^n = \frac{\rho}{1-\rho}$, and the derivative of it is $\sum_{n=1}^{\infty} n\rho^{n-1}$, we write $L$ as $(1-\rho)\rho\sum_{n=1}^{\infty} n\rho^{n-1}$. As $\rho < 1$,

$$L = (1-\rho)\rho\sum_{n=1}^{\infty} n\rho^{n-1} = (1-\rho)\rho\frac{d}{d\rho}(\frac{\rho}{1-\rho}) = \frac{\rho}{1-\rho}.$$

$$=> L = \frac{\lambda}{\mu-\lambda}$$

Using Little's Law ($L = \lambda W$), we get,

$$W = \frac{L}{\lambda} = \frac{1}{\mu-\lambda}$$

□

## 1.2 Multitype $M/M/1$ queue

In a multitype $M/M/1$ queue, each customer is of a type $t \in \mathcal{T} := \{1, \dots T\}$. The arrival rate of customer of type $t$ is a Poisson process with rate $\lambda(t)$ and the service time of the queue is exponential with rate $\mu$. For the queue to be stable, we must have $\rho := \sum_{t \in \mathcal{T}} \rho(t) < 1$, where $\rho(t) := \lambda(t)/\mu$.

We denote the state of queue by vector $\mathbf{t} = (t(1), t(2), \dots, t(n))$ where $n$ is the number of customers in queue and $t(p)$ is the type of customer at position $p \in \{1, \dots, n\}$.

Let $\mathbf{T} = \{\mathbf{t}_i : \mathbf{t}_i = (t(1), \dots, t(n))\,,\ t(a) \in \mathcal{T},\ a \in \{1, \dots, n\}\,,\ n \in \mathbb{N}_0\}$. The Markov chain recording the evolution of state has the transition probability function given by,

$$q(\mathbf{t}, \mathbf{t}') = \begin{cases} \lambda(t) & \text{if } \mathbf{t}' = (t(1), \dots, t(n), t) \\ \mu & \text{if } \mathbf{t}' = (t(2), \dots, t(n)) \end{cases}$$

The equilibrium distribution of the Markov Chain is

$$\pi(\mathbf{t}) = (1 - \rho) \prod_{a=1}^{n} \rho\left(t(a)\right)$$

## 1.3 Game Theory

**Definition 1.3.1.** A Strategic Game is a model in which 2 or more "players" or "Decision Makers" (DMs) each have a set of "actions" they can choose to perform. These players' actions are interlinked with each other in the way that the outcome for each player is also dependant on the choice of action by the other players. The model also consists of the preferences of each player based on the actions of all players, known as strategy profile.

|  | Ad | No Ad |
|---|---|---|
| Ad | 10000,10000 | 15000, 7000 |
| No Ad | 7000,15000 | 12000,12000 |

Table 1.1: Caption

**Example 1.3.2.** Two competing companies A and B selling the same product may choose to advertise their product or not. If A and B both advertise which includes the cost of advertising, then their earnings would be Rs.10000 per day which includes the cost of advertising. If only one of them advertises, then they will earn Rs.15000 while the other would get Rs.7000 earning. If none of them advertise, then both would save money on advertising and earn Rs.12000 each. The earnings or payoffs of their actions can be modeled as in table **??**.

This is an example of a strategic game where A and B have to decide the perform an action (Advertise or not) and the outcome depends on the actions of all players (A and B in this case).

**Definition 1.3.3.** For a strategic game with $N$ players, each having a set of actions $A_i (i \in \{1, 2, \ldots, N\})$, an action profile $a = \{a_1, a_2, \ldots, a_n\} \in A_1 \times A_2 \times \ldots \times A_N$ is said to be a *Nash Equilibrium* if, for each player $i$, $a_i$ is the best action for them given the actions of all other players.
The best action here is based on a payoff function $u_i : A_1 \times A_2 \times \ldots \times A_N \longrightarrow \mathbb{R}$ which each player $i$, has to maximize(or minimize).

In other words, for each player $i$, $a = \{a_1, a_2, \ldots, a_N\}$ is a Nash Equilibrium if

$$u_i(a) \geq u_i(a_{-i}, a'_i) \qquad \forall \ a'_i \in A_i$$

where $a_{-i} = \{a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_N\}$

**Definition 1.3.4.** If the player $i$ has some strategy $a_i$ such that for all possible strategies/actions of other players, $a_i$ fives the best possible outcome for $i$, then $a_i$ is a *dominant strategy* for $i$.

Precisely,

$$u_i(a_{-i}, a_i) \geq u_i(a_{-i}, a_i') \qquad \forall a_i' \in A_i, \forall a_{-i} \in A_1 \times \ldots \times A_{i-1} \times A_{i+1} \times \ldots \times A_N$$

**Definition 1.3.5.** Best response function For a player i, the best response function $B_i : A_1 \times A_2 \times \ldots \times A_{i-1} \times A_{i+1} \times \ldots \times A_n \to A_i$ is a function that takes the actions of all players other than $i$ and gives a set of actions from $A_i$ which give the maximum value of payoff function for $i$. So,

$$B_i(a_{-i}) = \{a_i \in A_i | u_i(a_i, a_{-i}) \geq u_i(a_i', a_i) \, \forall \, a_i' \in A_i\}$$

**Definition 1.3.6.** In pure strategy, each player has to choose an action from with probability 1. So a pure strategy, Nash equilibrium would be an action profile where each player has exactly one of their possible actions.

**Definition 1.3.7.** In mixed strategy games a player assigns a probability distribution to their set of actions. A Nash equilibrium would consist of the set of probability distributions on the actions of the players so that expected payoff of any player cannot be increased by a change in strategy by just them

# Chapter 2

# Queueing Network Games

## 2.1 Queueing Games

We consider a game on a network of nodes with players $\mathcal{N} = \{1, 2, \ldots, N\}$ and nodes $\mathcal{C} = \{1, 2, \ldots, C\}$. Every node $i \in \mathcal{C}$ is a FIFO queue with an exponential service rate $\mu_i$. Each player $j \in \mathcal{N}$ has its customers arriving with a Poisson process with rate $\lambda^{(j)}$. Every player $j \in \mathcal{N}$ is given a set of routes $R^{(j)}$, and must assign each of its customer to a route $r \in R^{(j)}$. The strategy of player $j \in \mathcal{N}$ is given by vector $p^{(j)}$, where $p_r^{(j)}$ denotes the probability that the player assigns its customers to route $r \in R^{(j)}$.

We consider each node $i \in \mathcal{C}$ to be a multi-type $M/M/1$ queue with customer types $\mathcal{T} = \{1, 2, \ldots, T\}$. For node $i \in \mathcal{C}$, the service time is exponential with rate $\mu_i$ and the arrival rate of customer of type $t \in \mathcal{T}$ is $\lambda_i(t)$.

The node $i \in \mathcal{C}$ is stable if and only if $\rho_i := \sum_{t \in \mathcal{T}} \rho_i(t) < 1$ where $\rho_i(t) = \lambda_i(t)/\mu_i$ for $t \in \mathcal{T}$.

**Example 2.1.1.** An example of a queueing game is given in Figure **??**. Here $\mathcal{N} = \{1, 2\}$, $\mathcal{R}^{(1)} = \{(1, 4), (2), (2, 3, 4)\}$ and $\mathcal{R}^{(2)} = \{(3), (1)\}$

## 2.2 Congestion Games

**Definition 2.2.1.** A Congestion Game is defined by tuple

$$\Gamma = (\mathcal{N},\, \mathcal{R},\, (\Sigma_i)_{i\in\mathcal{N}},\, (d_r)_{r\in\mathcal{R}})$$

where

- $\mathcal{N} = \{1, ..., n\}$ is the set of players
- $\mathcal{R}$ is the set of resources
- $\Sigma_j \subset 2^{\mathcal{R}}$ is the strategy space of player $j$
- $d_r : \mathbb{N} \rightarrow \mathbb{R}^+$ is the delay function for resource $r$.

The cost of player $j$ in state $S \in \Sigma_1 \times ... \times \Sigma_n$, is $c_j(S) = \sum_{r\in S_j} d_r(n_r(S))$ where $n_r = |\{j \in \mathcal{N} : r \in S_j\}|$.

**Definition 2.2.2.** A Weighted Congestion Game is defined by a tuple $\Gamma = (\mathcal{N},\, \mathcal{R},\, (\Sigma_i)_{i\in\mathcal{N}},\, (d_r)_{r\in\mathcal{R}},\, (w_i)_{i\in\mathcal{N}})$. Here

- $\mathcal{N}, \mathcal{R}, \Sigma_i$ have same definition as congestion game
- $d_r : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is a non-decreasing function giving the delay for using resource $r$
- $w_i$ is the weight of player $i$

The cost of player $j$ in state $S \in \Sigma_1 \times ... \times \Sigma_n$, is $c_j(S) = \sum_{r\in S_j} d_r(w_r)$ where $w_r := \sum_{\{i\in\mathcal{N}:r\in S_i\}} w_i$ is the sum of weights of players using resource $r$.

The following theorem shows that a Nash equilibrium exists for any congestion game and also shows that the algorithm in Chapter 3 converges to a Nash Equilibrium.

**Definition 2.2.3.** A step $S$ to $S' = (S'_i\ ,\ S_{-i})$ is called an improvement step if $c_i(S) > c_i(S')$.

**Theorem 2.2.4** (Rosenthal 1973)**.** *For congestion games, every sequence of improvement steps is finite.*

For proving this, we need to look at the Rosenthal's potential function, which for every state $S$ is defined to be

$$\Phi(S) = \sum_{r \in \mathcal{R}} \sum_{k=1}^{n_r(S)} d_r(k)$$

**Lemma 2.2.5.** *Let $S$ be any state and $S_i$ be an alternative strategy for player $i$. Then*

$$\Phi(S'_i, S_{-i}) - \Phi(S) = c_i(S'_i, S_{-i}) - c_i(S)$$

*Proof.* From definition of Rosenthal's Potential function

$$\begin{aligned} \Phi(S'_i, S_{-i}) - \Phi(S) &= \sum_{r \in \mathcal{R}} \sum_{k=1}^{n_r(S'_i, S_{-i})} d_r(k) - \sum_{r \in \mathcal{R}} \sum_{k=1}^{n_r(S)} d_r(k) \\ &= \sum_{r \in \mathcal{R}} \left( \sum_{k=1}^{n_r(S'_i, S_{-i})} d_r(k) - \sum_{k=1}^{n_r(S)} d_r(k) \right) \\ &= \sum_{r \in \mathcal{R}} \Delta_r \end{aligned} \tag{2.1}$$

where $\Delta_r := \sum_{k=1}^{n_r(S'_i, S_{-i})} d_r(k) - \sum_{k=1}^{n_r(S)} d_r(k)$.

- Case 1: $r \in S_i$ and $r \in S'_i$ then $\Delta_r = 0$
- Case 2: $r \notin S_i$ and $r \notin S'_i$ then $\Delta_r = 0$
- Case 3: $r \notin S_i$ and $r \in S'_i$ then $n_r(S'_i, S_{-i}) = n_r(S) + 1$ so $\Delta_r = d_r(n_r(S'_i, S_{-i}))$

- Case 4: $r \in S_i$ and $r \notin S'_i$ then $n_r(S'_i, S_{-i}) = n_r(S) - 1$ so $\Delta_r = -d_r(n_r(S))$

And we've

$$\begin{aligned} c_i(S'_i, S_i) - c_i(S) &= \sum_{r \in S'_i} d_r(n_r(S'_i, S_{-i})) - \sum_{r \in S_i} d_r(n_r(S)) \\ &= \sum_{r \in S'_i \; r \notin S_i} d_r(n_r(S'_i, S_{-i})) - \sum_{r \notin S'_i \; r \in S_i} d_r(n_r(S)) \\ &= \sum_{r \in \mathcal{R}} \Delta_r \qquad (2.2) \end{aligned}$$

From **??** and **??**, we got the desired result.

□

*Proof.* We prove Rosenthal's Theorem by contradiction. Assume that there exists an infinite sequence of improvement steps $S^{(1)}, S^{(2)}, \ldots$. By the previous Lemma and definition of improvement step

$$\Phi(S^{(i)}) > \Phi(S^{(j)}) \quad \forall\, i, j \in \mathbb{N},\ i < j \qquad (2.3)$$

As there are finitely many states, there must exist $i < j$ such that $S^{(i)} = S^{(j)}$. Hence $\Phi(S^{(i)}) = \Phi(S^{(j)})$ which contradicts the **??**. □

**Theorem 2.2.6.** *There exists a pure-strategy Nash equilibrium for $N$-player games on a network of single server queues when all customers have equal arrival rates i.e. $\lambda^j = \lambda$ for all $j \in \mathcal{N}$.*

*Proof.* We prove this by showing that this is equivalent to a congestion game. Since congestion games have a pure strategy Nash equilibrium, then so does this sub-class of queueing game. □

## 2.3 N-player game in discrete strategy space

For players with equal arrival rates, we can define a delay function at each node $i$.

$$d_i(x) = \frac{1}{\mu - x\lambda}$$

So the payoff/mean sojourn time for player $j$ in discrete-strategy profile $p$ is given by

$$f^{(j)}(p) = \sum_{i=1}^{C} \frac{\mathbb{1}_{i \in r_j(p)}}{\mu_i - x_i(p)\lambda} = \sum_{i \in r_i(p)} d_i(x_i(p))$$

where $x_i(p) = \sum_{j \in \mathcal{N}} \mathbb{1}_{i \in r_j(p)}$ and $r^{(j)}(p)$ is the route chosen by player $j$ in discrete-strategy $p$.

Clearly, $d_i(x)$ is a monotone increasing function in $x$. So this is a congestion game.

# Chapter 3

# Algorithm for computing Nash Equilibria of Queueing games

In this section, first we introduce and analyse the *best-response algorithm* as stated in Section 3 of [**?**].

In this section, we consider the problem of transforming a given model of a queueing network game into that of a traditional payoff matrix to be used for finding the Nash Equilibria, if any. As described in Chapter **??**, the model consists of a set of $N$ players and $C$ nodes and each player have $\lambda_j$ as the arrival rate of their customers, while each node has service rate $\mu_i$. $R^(j)$ are the sets of routes available to each player.

Here, we consider a discrete and finite strategy space for the players, so the payoffs to be computed will be that of *pure-strategy profiles*, where each cell of the payoff matrix will represent a distinct route chosen by each player.

When the above data is given as input, our algorithm outputs the payoff matrix in which each cell of the matrix gives the payoffs, which are the *Mean sojourn times* of the customers, of each player when following the route that the cell corresponds to. To find the Nash Equilibria, we can assign the goal for each player to be to minimize the mean sojourn time for their customers.

## 3.1 The Algorithm

The following python code returns the payoff matrix for N-player queueing games with C nodes in the queueing network, with a pure-strategy profile

Listing 3.1: Payoff Matrix

```
import numpy as np
from itertools import product

def read_floats():
    inp = input()
    inp = list(map(float, filter(lambda x: x!='', inp.split('␣'))))
    if len(inp) == 1:
        return inp[0]
    else:
        return tuple(inp)

N, C = map(int, read_floats())
service_rate = read_floats()
arrv_rate = [0]*N
routes = [[] for i in range(N)]

for j in range(N):
    num_routes, arr = read_floats()
    num_routes = int(num_routes)
    arrv_rate[j] = arr
    for _ in range(num_routes):
        #convert to 0 indexed int and remove the first element
        route = list(map(lambda x: int(x)-1, read_floats()))[1:]
        routes[j].append(route)
```

```
idx = 0
def calc_payoffs(S):
    global idx
    idx += 1
    freq_node = [0]*C
    for j in range(N):
        for i in range(len(routes[j])):
            for node in routes[j][i]:
                freq_node[node] += S[j][i]*arrv_rate[j]
    sojourn_node = [0]*C
    for i in range(C):
        assert(service_rate[i] > freq_node[i])
        sojourn_node[i] = 1/(service_rate[i] - freq_node[i])
    expected_waiting_time = [0]*N
    for j in range(N):
        for i in range(len(routes[j])):
            route_wait = 0
            for node in routes[j][i]:
                route_wait += sojourn_node[node]
            expected_waiting_time[j] += route_wait*S[j][i]
    return expected_waiting_time


payoff = np.zeros(tuple(list(len(routes[j]) for j in range(N)) + [N]))
indices = product(*tuple(range(len(routes[j])) for j in range(N)))
for index in indices:
    # index = (1,4,2,0,...) -> corresponding to choices of route
    S = [[0]*len(routes[j]) for j in range(N)]
    for j, choice in enumerate(index):
        S[j][choice] = 1
    payoff[index] = calc_payoffs(S)
```

```
print(payoff)
```

### 3.1.1 Input format

The code takes input as follows:

The first line contains two numbers: the number of players $N$ and number of nodes $C$.

The next line contains $C$ integers denoting the service rates of nodes.

Then for each player, the first line contains: number of routes $|R^{(j)}|$ and arrival rate $\lambda^{(j)}$.

Then then next $|R^{(j)}|$ lines contains: an integer $k$ followed by $k$ integers $r_1$ , $r_2$ …, $r_k$ which denotes a route $(r_1\ ,\ \dots\ ,\ r_k)$

### 3.1.2 Output format

The output is an $|R^{(1)}| \times |R^{(2)}| \times ... \times |R^{(N)}|$ matrix, where each cell is a tuple of size $N$, of Real numbers denoting the *mean sojourn time.*

*Remark* 3.1.1. The cell $(i_1\ ,\ i_2,\ \dots,\ i_n)$ gives the *payoffs/mean sojourn times* for the action profile $(r^{(1)}_{i_1}\ ,\ r^{(2)}_{i_2}\ \dots,\ r^{(N)}_{i_N})$. The $j^th$ value in the tuple gives the *mean sojourn time* for player $j$.

### 3.1.3 Examples

**Example 3.1.2.** *Input:*

```
2 4
5 7 6 9
3 2
```

```
1 2
2 1 4
3 2 3 4
2 2
1 1
1 3
```

This is a 2 player game with 4 nodes and the routes for each player are defined in the input in further lines. The following figure represents the network of queues and the possible routes for the players.

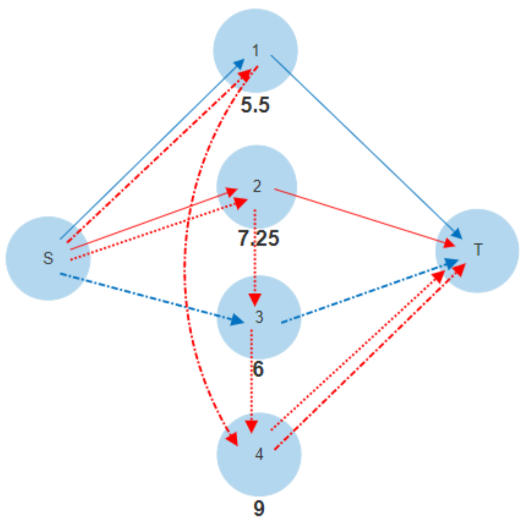

Figure 3.1: Two-player queueing game with four nodes

The red arrows correspond to player 1 and blue correspond to player 2. For multiple paths of the same player, arrows with a different dash pattern are used.

The result from this input is a 2 dimensional payoff-matrix with each cell being a tuple of 2 values, the *mean sojourn time* for each player.

*Output:*

| | |
|---|---|
| (0.19047619, 0.28571429) | (0.19047619, 0.25) |
| (0.80952381, 0.66666667) | (0.42857143, 0.25) |
| (0.58333333, 0.28571429) | (0.83333333, 0.5) |

Table 3.1: Example 1 - Payoff matrix

This is a $3 \times 2$ matrix, with each row in the $j^{th}$ dimension corresponding to a distinct route for player $j$.

**Example 3.1.3.** *Input:*

```
3 6

6 6 4.95 4.95 6 6

2 1

3 1 2 3

3 4 5 6

2 2

3 1 2 4

3 3 5 6

2 1

6 1 2 3 4 5 6

1 1
```

So this is a 3 player game with 6 nodes.

The following figure represents the network of queues and the possible routes for the players.

The red arrows correspond to player 1, blue correspond to player 2 and

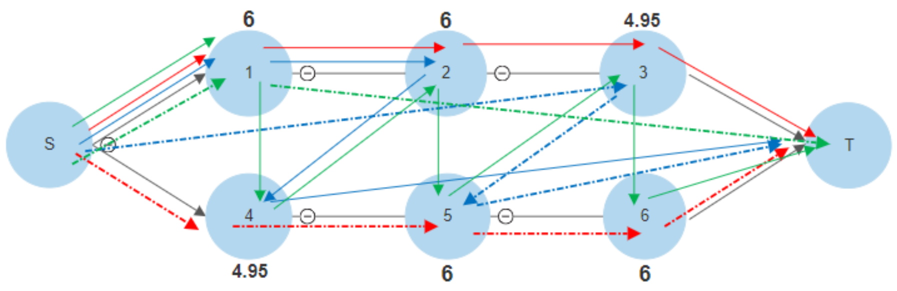

Figure 3.2: Three-player queueing game with six nodes

green correspond to player 3. For multiple paths of the same player, arrows with a different dash pattern are used.

The result from this input is a 3 dimensional payoff-matrix with each cell being a tuple of 3 values, the *mean sojourn time* for each player.

*Output:*

$$\begin{vmatrix} (1.34,\ 1.51,\ 2.25) & (1.09,\ 1.17,\ 0.50) \\ (1.55,\ 1.72,\ 2.47) & (0.96,\ 1.01,\ 0.25) \end{vmatrix} \qquad \begin{vmatrix} (1.55,\ 1.72,\ 2.47) & (0.91,\ 1.10,\ 0.33) \\ (1.34,\ 1.51,\ 2.25) & (0.92\ 1.01,\ 0.20) \end{vmatrix}$$

Table 3.2: Example 2 - Payoff matrix

This is a $2\times2\times2$ matrix, with each row in the $j^{th}$ dimension corresponding to a distinct route for player $j$.

# Chapter 4

# Conclusion and Future Extensions

## 4.1 Conclusion

In this paper, we have tried to emphasize on the application of game theory to optimize the routing strategy across a network of queues when, multiple rational Decision-makers are involved. Extensive research has been done in this field, especially in recent years; a good account of which can be found in [?], Chapter 8.

We have explored congestion games and the Rosenthal theorem (1973) which proves the finiteness of improvement steps in a congestion game. We have also studied the proofs for the existence of a pure-strategy Nash Equilibrium in an N-player queueing game with equal arrival rates.

We have introduced and illustrated with examples, an algorithm that gives the payoff matrix for general queueing games from which Nash Equilibrium can be computed. The existence of Nash Equilibrium need not guarantee the minimization of mean sojourn time for each player. Further topics of research are discussed in the next section.

## 4.2 Future Extension

After studying the existence of pure-strategy Nash-Equilibria for one of the subclasses in a discrete strategy space in [**?**], we would like to explore further for more subclasses and prove or disprove the existence of pure/mixed strategy Nash-Equilibrium for them. The subclassese include but are not limited to

- N-player games with equal arrival rate on multiserver queues ($M/M/c$ queues)
- 3-player games with identical service rates
- 2-player games with different distributions for the arrival and service rates of queues (like $M/E_k/1$, $M/D/1$) and addrd waiting cost in queues.
- Games with payoff functions for players other than their mean sojourn time (MST) like maximizing the MST for all other player or minimize waiting time at the final queue.
- $N$-player games with certain shapes of queueing networks without any constraint on arrival rate or service rate.