Optimal Resource Allocation and Optimal Investment Strategies

How to Minimise Costs and Maximise Profits

Report submitted in partial fulfilment of
the requirements for the degree:
MSc Complex Systems Modelling-
from Biomedical & Natural to Economic & Social Sciences
at King's College London

Department of Mathematics
King's College London
The Strand
London
WC2R 2LS
United Kingdom
18th September 2019

Abstract

At the large scales at which many industries operate at these days, the need for efficient resource allocation has become increasingly important. The manner in which to allocate such resources has been widely debated and a lot of research as been conducted into this.[1][2][3]. This paper will primarily be expanding upon the counter intuitive results attained in [4], where it was found that to reduce delay, you may be best to leave certain workers idle. We shall firstly generalise these results for a wider class of queuing and cost functions in such large systems we find that this phenomenon does in fact extend to these other applications, as well as observing more peculiarities in our solutions. Then we shall formulate this into an Economic Maximisation Problem, using these previously obtained allocations. We shall observe results in both a continuous setting, before then expanding to a discrete setting. We discover some interesting results as we alter various parameters, such as the maximum load the system can process. Our aim to reach an optimal solution where we can invest in units and allocate resources in a way that maximises our profits.

Contents

Chapter 1

Introduction

Allocating resources in the most efficient manner is on of the key aspects to project management. Deciding how much work you give to each worker can determine a whole range of things. How much profit will such a project make? What is the minimum time frame in which we can complete this project? And ultimately, whether or not the project will succeed or not. It is a problem of interest to a wide range of industries, be it office workers just allocating simple tasks or large scale computer mainframes handling thousands of terabytes of data. Here are some case studies of interest:

- **Public Transport:** Consider a scenario where you are a public transport provider. You have to serve a variety of different routes of different lengths and capacity. In order to provide such a service you will need to invest in a fleet of vehicles. the trick is deciding which vehicles and how many types of each vehicle you will invest in. for example, when it comes to buses, TfL operate ten types of bus [5], which all may have the same manual labour cost (one driver) but all have different capacities and operating costs depending upon what type of fuel they run on, which they need to optimise for what is of interest to them.

  This then opens up a wider question, depending on our objectives, what specific aspects of the system we want to optimise, which can often result in a conflict of interests. For example, a private provider's main aim is to make as much money from the system as possible, whereas a public provider will be trying their best to ensure there is minimum delay and the maximum number of passengers can be accommodated for across the system as a whole. Such differences in objectives has brought up a lot of debate regarding the public transport system in the UK, especially the privatisation of rail and it is up for debate whether such services should be made public again [6].

- **High Frequency Trading:** In the traditional Stock Market, the buying and selling of stocks was conducted on busy trade floors with loud traders directing their order at one another these. Modern developments have the rise of the internet and advances in technology, these trades are often conducted by computers, using complex algorithms capable of performing many trades within a single second. This process is known as *High Frequency Trading (HFT)*. This a process which relies on the speed of the whole process allowing the completion of millions of transactions with small marginal profits in order to gain the most from the system. [7]. Gaining even a millisecond advantage over competitors can result in an extreme growth in profits. If we are given a reliable model for how a stock will behave in the future, optimised allocations will take full advantage of this.

- **Computer Processors**: Online computer servers are responsible transfer large quantities of data across large scale systems. They need to be able to transfer appropriate amounts of information in an as efficient way as possible to not only minimise the delay across the system but also maximise profits of the service provider.

These situations present two main problems to solve:

- **Resource Allocations:** This is where we look to optimise the *allocation* of our load across the whole system. That is how much work we provide each server with. We look to choose an allocation such that we are able to minimise our overall or average costs across our system This cost may be in the form of delay or computational cost to process the load.

- **Investment Strategies:** Once we have our optimal allocation, we now wish to have an optimal set of servers to accommodate for this allocation. We need to invest in servers in such a way that we can minimise our total operating costs across all servers and maximise the profits obtained from the system.

The aim of this paper is to develop strategies to tackle these problems using many aspects of queuing theory and a wide range of optimisation techniques. We shall firstly focus upon the mathematical framework and look to generalise already established results. This shall be done to produce mathematical models for a range of problems which we shall aim to solve.

Chapter 2

Literature Review and Theoretical Background

2.1 Queuing Theory

Queuing Theory is an area of mathematics often used in Operational Research looking into *queues*[8]- a process which require a set time *waiting* and a set time being *served* before the process can be completed. It originates from the research of Agner Krarup Erlang, a Danish Engineer who used it when he set about creating models for the Copenhagen Telephone Exchange.[9] Erlang noted that queues can exhibit phase-transition like behaviour and how the behaviour of the queue is determined heavily by the arrival and service rates of the system. Since then, these models have been used for a wide variety of useful applications, such as modelling the spread of disease in epidemiology[10][11].

So what exactly is a **Queuing System**? The premise of these models is follows the following intuition. Imagine when you got to shop to by something. Once we have decided what we want to buy, we have to wait in a queue to get served, then get served and then we can leave the system. The *Queuing System* is known as the period where you are waiting in the queue and being served. This is the part of the system that the shop can optimise to either reduce Delay time or increase the number of customer they are able to serve in a certain time period.

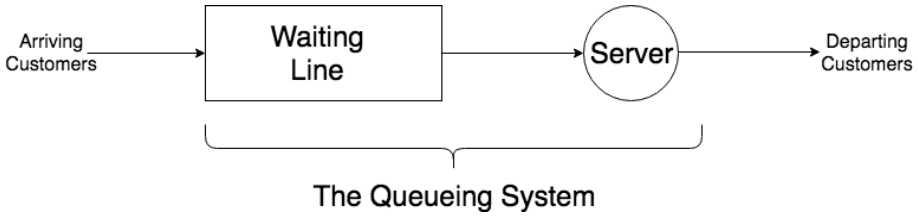

Figure 2.1: A diagram depicting the Queuing System- with customers entering the system into the waiting line, proceeding to get served before finally exiting the system

This basic concept has been expanded upon further in order to account for various factors such as the number of servers, the distribution of arrival service rate and the capacity of the queue.

The aim of the paper is to focus on using these models to determine the total time spent in the system, as well as the costs they may incur. In order to do so, we will need to focus on some key results as well as the foundations of Queuing Theory. When it comes to queuing theory, the two main quantities to consider are:

- λ: The Arrival Rate into the System. This determines the flow *into* the system and is usually a random variable following a specified distribution.
- μ: The Service Rate of the server of the System. This controls the flow *out* of the system and like with the Arrival Rate, this too is a random variable following a distribution
- $\rho = \frac{\lambda}{\mu}$: The Load of the System. In order for the system to remain stable, we need to ensure that the load remains less than 1. Otherwise, if this bound is exceeded we don't have a stable system. When we say that a system is stable, this implies that we can always progress to a scenario where there is no longer a queue have a queue. So a load exceeding that exceed 1 implies the queue never empties- which is a very undesirable scenario to be avoided.[12]

With these quantities we are able to predict a variety of properties of the model. As previously said, we want to focus on minimising the total time spent in the system. To do this we need to refer to two of the main results in Queuing Theory: **Little's Law**[13][14] and the **Pollaczek–Khinchine Formula**[15] [16]:

Theorem 2.1 (Little's Law)**.** *For a queue with an average arrival rate of λ and the average time spent in the system is W, the average number in the system, L, is given by:*

$$L = \lambda W \tag{2.1}$$

Theorem 2.2 (Pollaczek–Khinchine Formula)**.** *The formula states that the mean queue length, L, is given by:*

$$L = \rho + \frac{\rho^2 + \lambda^2 \operatorname{Var}(\mu^{-1})}{2(1-\rho)} \tag{2.2}$$

By combining these two formulae together we see that the total time spent in the system is given by:

$$W = \frac{\rho + \lambda\mu \operatorname{Var}(\mu^{-1})}{2(\mu - \lambda)} + \mu^{-1} \tag{2.3}$$

In order to model a variety of systems, we have different assumptions for each system and these are signified with the use of *Kendall's Notation*[17]. This tells various properties of the queue, such as the distribution of service and arrival rates, the capacity of the queue, the number of servers in the system and so on. For example, the most simple model, the $M/M/1$ queue signifies a single server queue with an exponential arrival and service rates.

2.2 Optimisation Methods

When it comes to optimising such processes, there are a variety of techniques used such as linear and geometric programming [18]. We shall go into detail the main methods which will be used in this paper, firstly in a continuous setting, then that of a discrete setting.

2.2.1 Continuous Optimisation

A very popular method of convex optimisation is the usage of the **Lagrange Multiplier Method** [19]. It is used to optimise some objective functions in respect to various constraints composed on the system. This involves solving a system of the form:

$$\begin{aligned} &\text{Maximise/Minimise} \\ &\qquad f(x) \\ &\text{in respect to} \\ &\qquad g_i(x) = 0 \quad \forall i \end{aligned}$$

However this only works for when we have strictly tight constraints. We may have a system where we are dealing with inequality constraint of the form:

$$\begin{aligned} &\text{Maximise/Minimise} \\ &\qquad f(x) \\ &\text{in respect to} \\ &\qquad g_i(x) = 0 \quad \forall i \\ &\qquad h_j(x) \leq 0 \quad \forall j \end{aligned}$$

This where we have to use what is know as the **Karush Kuhn Tucker Conditions**, a set of conditions used to optimise such systems, firstly proved in the Master's thesis of William Karush in 1939 but published first by Harold W. Kuhn and Albert W. Tucker in 1951[20][21].

Theorem 2.3 (Karush Kuhn Tucker Conditions)**.** *Suppose that the* ***objective function*** $f : \mathbb{R}^n \to \mathbb{R}$ *and the constraint functions* $g_i : \mathbb{R}^n \to \mathbb{R}$ *and* $h_j : \mathbb{R}^n \to \mathbb{R}$ *are continuously differentiable at a point* x^**. If* x^* *is a local optimum and the optimisation problem satisfies some regularity conditions, then there exist constants* μ_i $(i = 1, \ldots, m)$ *and* λ_j $(j = 1, \ldots, \ell)$*, called KKT multipliers, such that:*

- *Stationarity:*

  For maximising $f(x)$*:*

  $$\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*),$$

- *Primal feasibility:*

  $$g_i(x^*) \leq 0, \text{ for } i = 1, \ldots, m$$
  $$h_j(x^*) = 0, \text{ for } j = 1, \ldots, \ell$$

- *Dual feasibility :*

$$\mu_i \geq 0, \ for\ i = 1, \ldots, m$$

- *Complementary slackness :*

$$\mu_i g_i(x^*) = 0, \ for \ \ i = 1, \ldots, m.$$

This is what was primarily used to obtain the solutions in [4], and we too shall be using it to obtain our own solutions. This has proven to be a popular method in economics, as it enables to optimise problems without exhausting all of our resources[22].

2.2.2 Discrete Optimisation

In this paper, we shall also encounter cases where the units we are dealing with are not entirely continuous. As a result, methods like the KKT conditions may not translate well when optimising for discrete cases. So instead different methods are required to optimise functions. These may include Discrete Hamiltonian's[23] or plane cutting[24]. However we shall be using a more algorithmic approach similar to **Simulated Annealing.**

Simulated annealing algorithm

```
1  Select the best solution vector x0 to be optimized
2  Initialize the parameters: temperature T, Boltzmann's constant k, reduction factor c
3  while termination criterion is not satisfied do
4      for number of new solution
6          Select a new solution: x0+Δx
7              if f(x0+Δx) > f(x0) then
8                  fnew = f(x0 + Δx);  x0 = x0 + Δx
9                  else
10                     Δf = f(x0 + Δx) − f(x0)
11                     random r(0, 1)
12                         if r > exp(−Δf/kT) then
13                             fnew = f(x0 + Δx),  x0 = x0 + Δx
14                             else
15                             fnew = f(x0),
16                         end if
17             end if
18         f = fnew
19         Decrease the temperature periodically: T = c x T
20     end for
21 end while
```

Figure 2.2: Pseudo-Code for the Simulated Annealing algorithm[25]

It is a method which was first proposed in [26]. Annealing is the process of repeatedly heating and cooling a material till it reaches it's molecules reach an "*Optimal Configuration*". Whilst initially just used to reach the ground state (minimal energy in its solid state) of a material, it has also become a very useful tool in discrete optimisation and help avoid large combinatorial problems[27].

2.3 Optimal Routing Policy

Now that we have covered the groundwork we require to tackle our problem, we shall be summarise a paper [4] which deals with the first half of our problem, Resource Allocation. In this paper, the authors look to minimise the average delay time (The time spent in the system) across a network of N servers, each modelled by the previously discussed $M/M/1$ queue. For each individual server i, we have a delay of:

$$D_i = \frac{1}{\mu_i - \lambda_i}$$

The paper aims to optimise the average delay time across the whole network by minimising the objective function:

$$D = \frac{1}{\lambda}\sum_{i=1}^{N}\lambda_i D_i = \frac{1}{\lambda}\sum_{i=1}^{N}\frac{\lambda_i}{\mu_i - \lambda_i}$$

in respect to the constraints:

$$\lambda = \sum_{i=1}^{N}\lambda_i$$

$$\lambda_i > 0 \quad i = 1, 2, .., N$$

By using Lagrangians and the KKT Conditions, they attained the following optimal service rate:

$$\lambda_i^* = \max\left(\mu_i - \sqrt{\frac{\mu_i}{\lambda\phi_0}}, 0\right)$$

Which gave this optimal delay rate:

$$D_{\text{OPT}} = \frac{1}{\lambda}\sum_{i=1}^{N}(\sqrt{\lambda\phi_0\mu_i} - 1)\Theta(\sqrt{\lambda\phi_0\mu_i} - 1)$$

It is difficult to conceptualise the difference analytically. However, we can derive a solution numerically. These reproducible result is shown in Figure 2.3. The main very counter intuitive result of this paper is the high number of idle resources for a wide range of loads. You some how reduce time spent in the system by giving servers no work to do! This is an interesting result worth expanding upon and generalising for a wider range of contexts, including the Economic Extension suggested at the end of the paper.

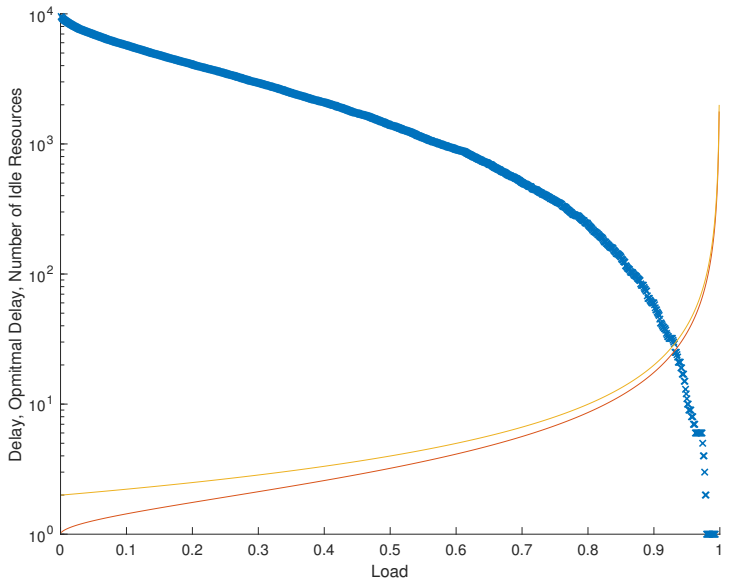

Figure 2.3: A graph depicting the Number of Idle Servers in the System(Blue), the Standard Delay (Orange) and the Optimised Delay (Red) with a set of Uniformly Distributed Servers

2.3.1 Similar Problems and further literature review

Upon further research into the issues regarding the applications of this problem, there were a variety of interesting studies done into various aspects, but nothing quite answers the problems of both resource allocation and investment strategies using idle resources. These has been plenty of work done looking into minimising delay times across wide scale networks, [28][18]. A wide variety of queuing functions have been looked into, including the $G/G/1$ queue [29], which is the most generalised for of an $M/M/1$ queue. They have looked at minimising the total delay rather than the average delay as seen in [4]. However the result of [4], where we allocate *no resources* at all to certain servers, has seemingly yet to have been replicated. So for completion, it is worth generalising the previous model for more general queues to provide a link in this literature.

In regards to Investment strategies, the usage of queuing theory [30] and wide scale networks [31] have been used in industrial economics for some time now. For example, [32] looks into using the simulated annealing algorithm in order to choose which processing units should be included into a system.

When it comes to what we do on this paper, based upon what has already been achieved, due to the lack of follow up work on the approach taken in [4], which took research into a unique direction, we should continue along this path developing it further, with the help of already known theories and by the end, we should be at a stage where we can then link this all back up with already established work.

Chapter 3

Optimal Resource Allocation

3.1 Altering the distribution of servers

The first extension on [4], is to see what differs as we change the distribution of μ_i's across the whole system, and how the results differ as we alter the parameters. In [4], the servers were uniformly distributed in the numerical results, so it would be interesting if these results alter if the distribution were to be changed. We decided to use an Exponential distribution and a Gamma distribution. The results can be observed in the following figures:

- **Exponentially Distributed Servers:** The first thing to notice here is that the curve for idle servers is a lot smoother, reflecting the exponential nature of their distribution. However, as we change the mean of the distribution, there is no change in the number of idle servers and the delay times just scale down in respect to this.

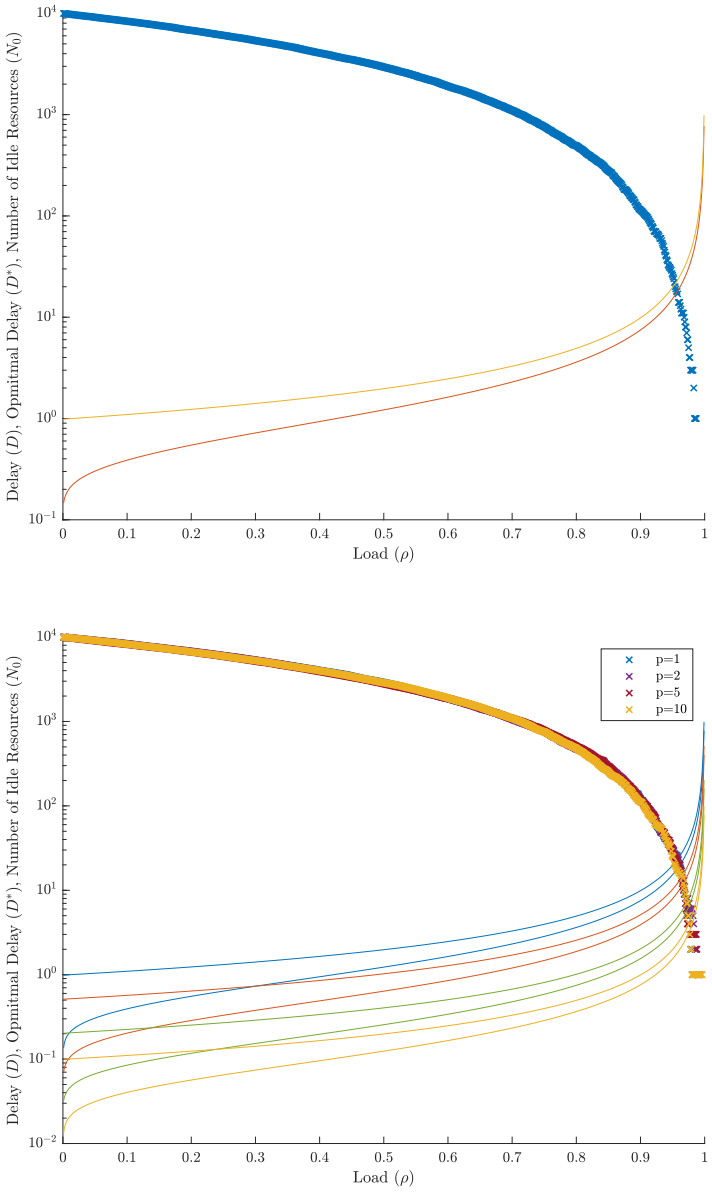

Figure 3.1: The first graph (Top) indicates how the delay and distribution of idle servers vary if $\mu_i \overset{i.i.d}{\sim} \text{Exp}(1)$ and the second graph shows the same for $\mu_i \overset{i.i.d}{\sim} \text{Exp}(p)$ for $p = 1, 2, 5, 10$

- **Gamma Distributed Servers:** Because of the relation to the exponential distribution, the Gamma Distribution shares a very similar shape. However, as we change the shape parameter of the distribution, we see that the number of idle servers in fact decreases. This is because, as you increase the parameter, the distribution becomes more sharper around a certain area, thus the distribution of servers become a lot more uniform and concentrated around the mean, causing the idle resource curve to 'collapse', as the powers of the resources are a lot closer together, resulting in less 'comparatively powerful' resources.

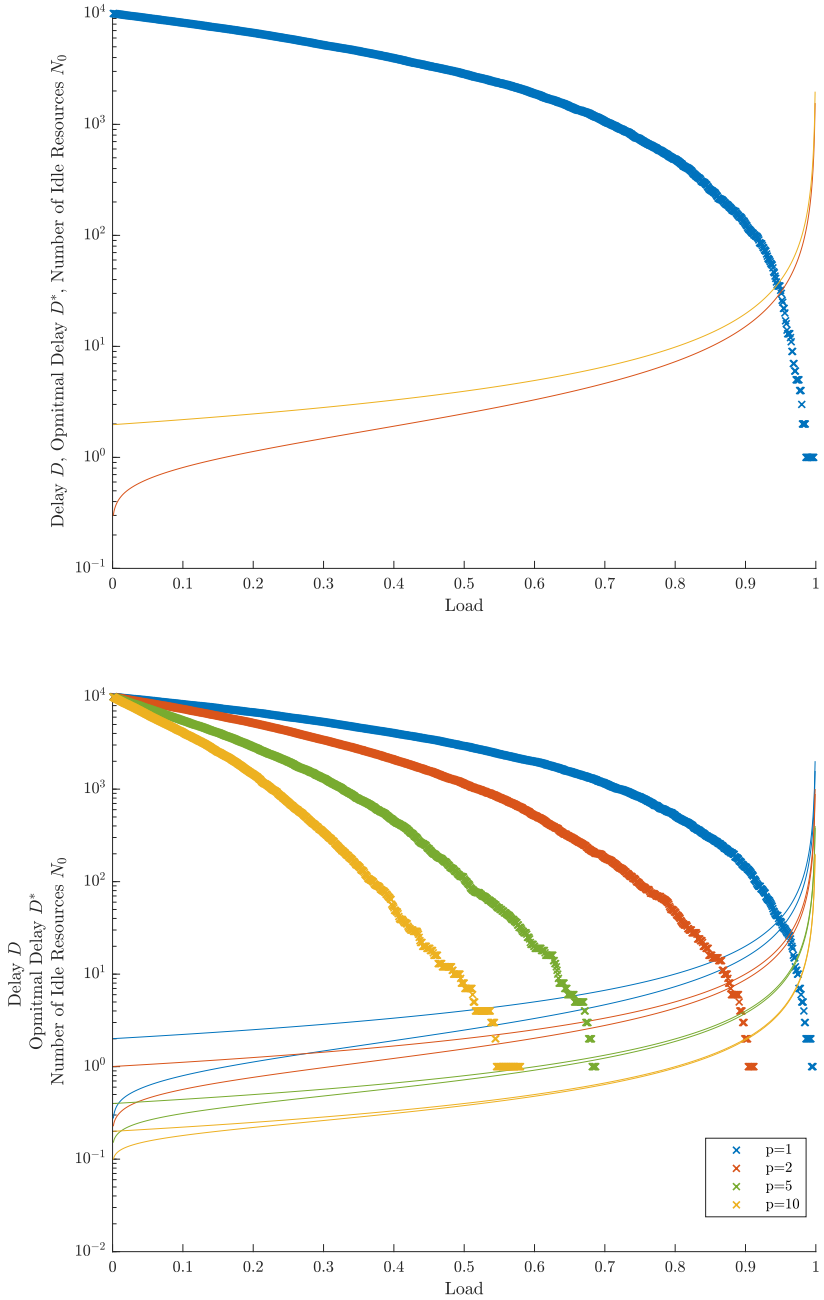

Figure 3.2: The first graph (top) indicates how the delay and distribution of idle servers vary if $\mu_i \overset{i.i.d}{\sim} \text{Gamma}(1, \frac{1}{2})$ and the second graph shows the same for $\mu_i \overset{i.i.d}{\sim} \text{Gamma}(p, \frac{1}{2})$ for $p = 1, 2, 5, 10$

3.2 Generalising to different Delay Functions

Next we will look to generalise the result on a system with a wider class of queuing systems. The $M/M/1$ queue looks at a queues with an exponential arrival and service rate. In our previous system, we assumed all the servers followed this discipline, but there are other restrictions we can put onto these queue. We shall be generalising the result for other distributions, with their own delay functions, and see if the result still holds up.

- $M/D/1$ Queues- This a queue with a *deterministic* service rate (It is fixed and not treated as a random variable)
$$D_i = \frac{1}{\mu_i} + \frac{\lambda_i}{2\mu_i(\mu_i - \lambda_i)} \tag{3.1}$$
- $M/G/1$ Queues- This is the overall generalisation of the service rate and provides the delay given the service rate, μ_i, follows any given distribution. This in fact reduces to from the Pollaczek-Khinchin formula and Little's Law from equation (2.3):
$$D_i = \mathbb{E}[\mu_i^{-1}] + \frac{\lambda_i \mathbb{E}[\mu_i^{-2}]}{2(1-\rho_i)} \tag{3.2}$$

3.2.1 Generalising the methodology

Before doing that, let's generalise the original optimisation problem:

$$\text{Minimise} \quad D = \frac{1}{\lambda}\sum_{i=1}^{N} \lambda_i D_i \quad \text{where} \quad D_i = \frac{f(\rho_i)}{\mu_i}$$

In respect to the constraints:

$$\lambda = \sum_{i=1}^{N} \lambda_i$$
$$\lambda_i \geq 0 \quad i = 1, 2, .., N$$
$$\lambda_i \leq \mu_i \quad i = 1, 2, .., N$$

Notice the introduction of the new constraint $\lambda_i \leq \mu_i$. This is to ensure no server is dealing with overcapacity. This constraint isn't needed for the $M/M/1$ delay function, since it diverges at maximum load, and can be omitted in such cases. These equations form the following Lagrangian:

$$\mathcal{L}(\lambda_i, \psi_i, \phi_i, \phi_0) = D - \phi_0(\lambda - \sum_{i=1}^{N} \lambda_i) - \sum_{i=1}^{N} \phi_i \lambda_i - \sum_{i=1}^{N} \psi_i(\mu_i - \lambda_i) \tag{3.3}$$

Note that $\lambda_i D_i = \frac{\lambda_i}{\mu_i} f(\rho_i) = g(\rho_i)$ This Lagrangian provides us with the KKT Conditions:

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda}\frac{\partial g}{\partial \lambda_i} - \phi_0 - \phi_i + \psi_i = 0 \quad i = 1, 2, .., N$$

$$\frac{\partial \mathcal{L}}{\partial \phi_0} = \lambda - \sum_{i=1}^{N} \lambda_i = 0$$
$$\phi_i \lambda_i = 0 \quad \lambda_i \geq 0 \quad i = 1, 2, .., N$$
$$\psi_i[\mu_i - \lambda_i] = 0 \quad \mu_i \geq \lambda_i \quad i = 1, 2, .., N$$

Now consider the three possible cases:

1. The Server is idle: $\lambda_i = 0$, $\phi_i \neq 0$ and $\psi_i = 0$
2. The Server is doing work: $\lambda_i \in (0, \mu_i)$, $\phi_i = 0$ and $\psi_i = 0$
3. The Server is working at maximum load: $\lambda_i = \mu_i$, $\phi_i = 0$ and $\psi_i \neq 0$

These reduce $\frac{\partial \mathcal{L}}{\partial \lambda_i}$ into the following cases:

1.
$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda}\frac{\partial g}{\partial \lambda_i} - \phi_0 - \phi_i = 0$$

2.
$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda}\frac{\partial g}{\partial \lambda_i} - \phi_0 = 0$$

3.
$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda}\frac{\partial g}{\partial \lambda_i}(1) - \phi_0 + \psi_i = 0$$

Equations 1 and 3 provide you with the values of the KKT Multipliers ϕ_i and ψ_i, but these are of little interest to us. Equation 2 is what we are interested in and we solve it in respect to λ_i. We shall denote this solution as $\lambda_i^o(\phi_0, \mu_i, \lambda)$. Thus this results in the optimal load allocation being as follows:

$$\lambda_i^* = \begin{cases} 0 & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \leq 0 \\ \lambda_i^o(\phi_0, \mu_i, \lambda) & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \in (0, \mu_i) \\ \mu_i & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \geq \mu_i \end{cases} \tag{3.4}$$

In order to retrieve the Lagrangian Multiplier, ϕ_0, we substitute the solutions for λ_i^* into the constraint:

$$\lambda = \sum_{i=1}^{N} \lambda_i \tag{3.5}$$

$$= \sum_{i=1}^{N} \lambda_i^o(\phi_0, \mu_i, \lambda)\Theta(\lambda_i^o(\phi_0, \mu_i, \lambda))\Theta(\lambda_i^o(\phi_0, \mu_i, \lambda) - \mu_i) + \mu_i\Theta(\mu_i - \lambda_i^o(\phi_0, \mu_i, \lambda)) \tag{3.6}$$

This equation can be solved numerically. This now gives us everything to obtain an optimal delay function:

$$D_{OPT}^* = \sum_{i}^{N} g\Big(\frac{\lambda_i^o(\phi_0, \mu_i, \lambda)}{\mu_i}\Big)\Theta(\lambda_i^o(\phi_0, \mu_i, \lambda))\Theta(\lambda_i^o(\phi_0, \mu_i, \lambda) - \mu_i) + g(1)\Theta(\mu_i - \lambda_i^o(\phi_0, \mu_i, \lambda)) \tag{3.7}$$

The number of units which are Idle and at Full Capacity are given by:

$$N_0 = \sum_{i=1}^{N} \Theta(-\lambda_i^o(\phi_0, \mu_i, \lambda)) \qquad N_F = \sum_{i=1}^{N} \Theta(\lambda_i^o(\phi_0, \mu_i, \lambda) - \mu_i) \tag{3.8}$$

We shall now use this to find the optimal resource allocation of the previously discussed queues as well as other other potential candidate functions to be minimised.

3.3 Using a Quadratic Delay/Cost Function

Before handling any of the *real queues*, it is worth testing out the above methodology on a generalised cost function, in the form of a quadratic. This is of use for us, as we find later many of the other formulae break down into a quadratic equation to be solve. Hence, let our delay function be:

$$D_i = \frac{1}{\mu_i}[d_0 + d_1\rho_i + d_2\rho_i^2]$$

This gives the following form for $g(\rho_i)$:

$$g(\rho_i) = d_0\rho_i + d_1\rho_i^2 + d_2\rho_i^3$$

$$\frac{\partial g}{\partial \lambda_i} = \frac{d_0}{\mu_i} + \frac{2d_1\rho_i}{\mu_i} + \frac{3d_2\rho_i^2}{\mu_i}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda\mu_i}[d_0 + 2d_1\rho_i + 3d_2\rho_i^2] - \phi_0 + \psi_i - \phi_i = 0$$

Apply KKT to obtain the explicit value of the KKT Multipliers:

$$(i)\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{d_0}{\lambda\mu_i} - \phi_0 - \phi_i = 0 \quad \implies \phi_i = \frac{d_0}{\lambda\mu_i} - \phi_0 \tag{3.9}$$

$$(iii)\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda\mu_i}[d_0 + 2d_1 + 3d_2] - \phi_0 + \psi_i = 0 \quad \implies \psi_i = \phi_0 - \frac{1}{\lambda\mu_i}[d_0 + 2d_1 + 3d_2] \tag{3.10}$$

Apply KKT to obtain an optimal value of ρ_i:

$$(ii)\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda\mu_i}[d_0 + 2d_1\rho_i + 3d_2\rho_i^2] - \phi_0 \tag{3.11}$$

$$\implies 3d_2\rho_i^2 + 2d_1\rho_i + d_0 - \lambda\phi_0\mu_i = 0 \tag{3.12}$$

By solving this quadratic, we attain the optimal resource allocation:

$$\boxed{\lambda_i^o(\phi_0, \mu_i, \lambda) = \frac{-d_1 + \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}}{3d_2}\mu_i}$$

Thus:

$$\lambda_i^* = \begin{cases} 0 & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \leq 0 \\ \frac{-d_1 + \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}}{3d_2}\mu_i & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \in (0, \mu_i) \\ \mu_i & \text{if } \lambda_i^o(\phi_0, \mu_i, \lambda) \geq \mu_i \end{cases}$$

In order to obtain the Lagrange Multiplier ϕ_0, substitute this result into the Equation (3.7) from the previous section. Our optimised delay is now:

$$D^* = \frac{1}{\lambda}\sum_{i=1}^{N}\lambda_i[d_0 + d_1\rho_i^* + d_2\rho_i^{*2}] \tag{3.13}$$

$$= \tfrac{1}{\lambda}\textstyle\sum_{i=1}^{N} \frac{-d_1 + \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}}{3d_2}\Theta\Big(-d_1 + \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}\Big)\Theta\Big(3d_2 + d_1 + \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}\Big) \tag{3.14}$$

$$+[d_0 + d_1 + d_2]\Theta\Big(\sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)} - d_1 - 3d_2\Big) \tag{3.15}$$

The Number of resources which are either idle or operating at full capacity are given by:

$$N_0 = \sum_{i=1}^{N}\Theta\Big(d_1 - \sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)}\Big) \tag{3.16}$$

$$N_F = \sum_{i=1}^{N}\Theta\Big(\sqrt{d_1^2 - 3d_2(d_0 - \lambda\phi_0\mu_i)} - d_1 - 3d_2\Big) \tag{3.17}$$

3.3.1 Numerical Results

Likewise with the result from [4], this solution is indeed optimal and yet again we have set amount of Idle resources for low loads. However, due to the bounded nature of the Quadratic function, there is also a region of servers operating at full capacity. In fact, we have a regime present for loads exceeding 0.6, where we have a collection of servers which are both idle *and* running at full capacity. This suggests there to be some inefficiency with the servers we have present and may suggest that there could be a better choice of what servers to have, depending on the load the system is expecting to receive. However, despite this being the case, this still gives us an optimal solution, further expanding upon this idle worker paradox.

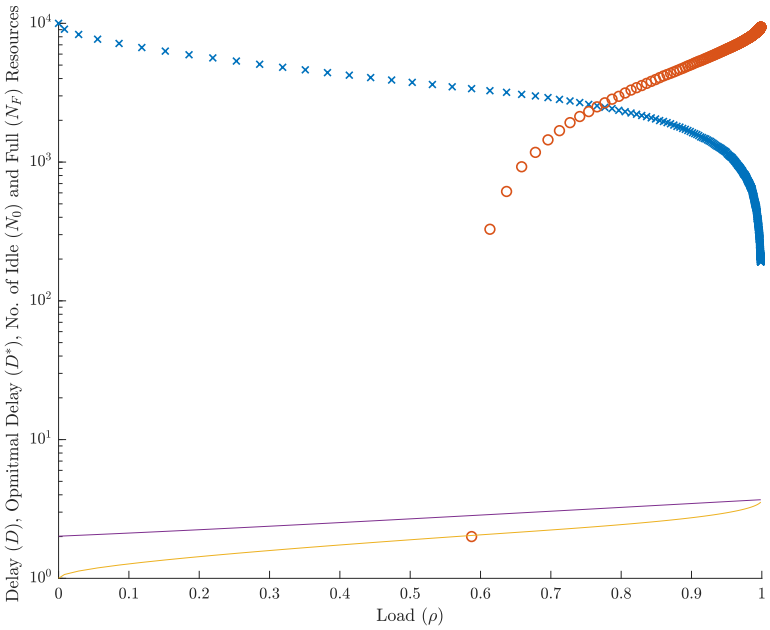

Figure 3.3: A depicting the number of Idle (Blue) and Full (Red) servers in the system as well as the optimised (Yellow) and Ad-Hoc (Purple) delay functions.

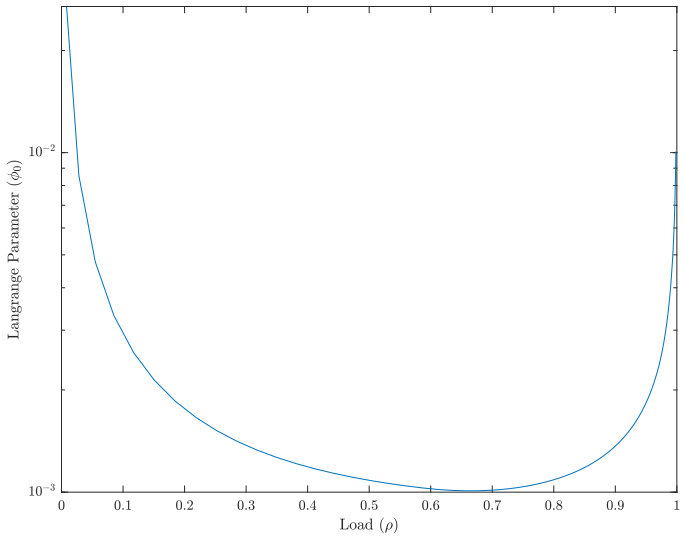

Figure 3.4: A graph depicting how the value of the Lagrange Multiplier, ϕ_0, alters as we change the load.

3.3.2 Changing the distribution

If we change the distribution of servers, we get some interesting results. Figure 3.5 replicates the numerical results shown in Figure 3.3, but this time sing Exponentially and Gamma distributed servers.

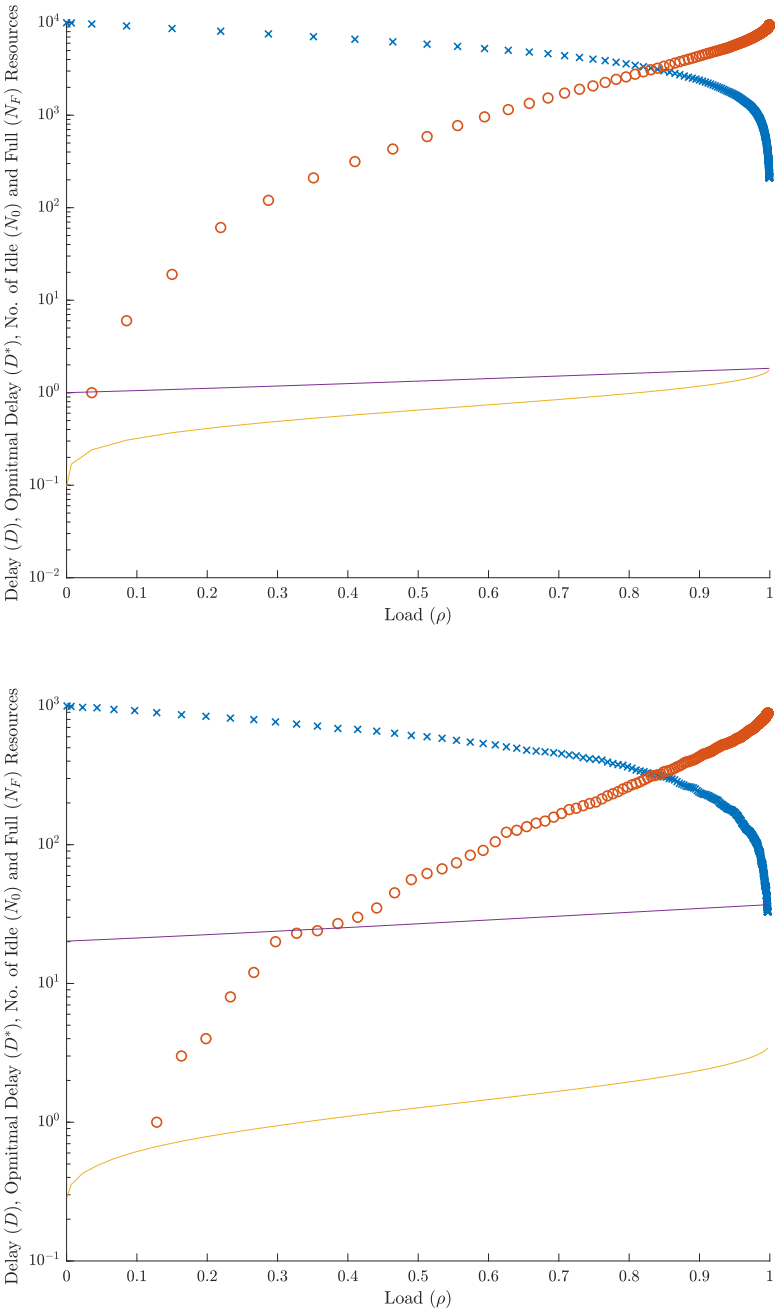

Figure 3.5: The first graph (top) indicates how the delay and distribution of idle servers vary if $\mu_i \overset{i.i.d}{\sim} Exp(1)$ for and the second graph shows the same for $\mu_i \overset{i.i.d}{\sim} \text{Gamma}(1, 0.5)$

If we to have a set of exponentially distributed servers, we notice even with lower loads, there will already be a set amount of full servers, so the number of full servers steadily increases with the load. In the case of Gamma distributed servers, we notice a similar pattern in regard to idle servers, but also note how the optimal delay time is now considerably lower than the Ad Hoc Delay time. This is perhaps not only to do with the exponential nature but also the shape of the distribution having a sharper peak around a certain region.

3.4 Queues with Exponential Delay Functions

Now we shall look at a a Queue with an Exponential Delay function:

$$D_i = \frac{A_0}{\mu_i} e^{A_1 \rho_i}$$

This gives the following form for $g(\rho_i)$:

$$g(\rho_i) = A_0 \rho_i e^{A_1 \rho_i}$$

$$\frac{\partial g}{\partial \lambda_i} = \frac{A_0}{\mu_i} e^{A_1 \rho_i} (1 + A_1 \rho_i)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{A_0}{\lambda \mu_i} e^{A_1 \rho_i} (1 + A_1 \rho_i) - \phi_0 - \phi_i + \psi_i = 0$$

Apply KKT to obtain the explicit value of the KKT Multipliers:

$$(i) \frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{A_0}{\lambda \mu_i} - \phi_0 - \phi_i = 0 \quad \implies \phi_i = \frac{A_0}{\lambda \mu_i} - \phi_0 \tag{3.18}$$

$$(iii) \frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{A_0}{\lambda \mu_i} e^{A_1} (1 + A_1) - \phi_0 + \psi_i = 0 \quad \implies \psi_i = \phi_0 - \frac{A_0}{\lambda \mu_i} e^{A_1} (1 + A_1) \tag{3.19}$$

Apply KKT to obtain an optimal value of ρ_i:

$$(ii) \frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{A_0}{\lambda \mu_i} e^{A_1 \rho_i} (1 + A_1 \rho_i) - \phi_0 \tag{3.20}$$

$$\implies e^{A_1 \rho_i} (1 + A_1 \rho_i) = A_0^{-1} \lambda \phi_0 \mu_i \tag{3.21}$$

This is a fixed point equation left for us to solve:

$$\rho_i = A_1^{-1} \left(A_0^{-1} \lambda \phi_0 \mu_i e^{-A_1 \rho_i} - 1 \right) \tag{3.22}$$

This needs to be solved numerically in conjunction with the identity:

$$\lambda = \sum_{i=1}^{N} \lambda_i$$

In order to obtain our optimal allocation, λ_i^*, as well as the Lagrange Multiplier, ϕ_0. Whilst we won't go on to solve this system here, we have shown that this is yet another system for which an optimal result can be obtained for.

One thing to note, regarding the quadratic and exponential functions, is that they're not exactly a *realistic* delay functions. This is because the nature of queues normally results in a divergence as $\rho \to 1$, in order to maintain stability, and thus don't need to account for any additional constraints to account for this. However, think of them as cost functions that a firm may incur upon a process which they are running. These functions shall be returned to in the next chapter. But first, we shall return to solving the problem with the standard delay formulae mentioned previously for the $M/D/1$ and $M/G/1$ queues.

3.5 The $M/D/1$ queue

Now we look to the case of the $M/D/1$ Queue, This is very similar to the $M/M/1$ queue, but instead of having each μ_i treated as an exponentially distributed random variable, we have it being treated as a constant.[1] While this may not seem like it will cause an issue, it does since a $M/D/1$ has a different Delay function:

$$D_i = \frac{1}{\mu_i} + \frac{\lambda_i}{2\mu_i(\mu_i - \lambda_i)} \tag{3.23}$$

The optimisation problem becomes to solve the following equation, subject to the usual constraints.

$$\frac{1}{\lambda}\sum_{i=1}^{N} \lambda_i D_i = \frac{1}{\lambda}\sum_{i=1}^{N} \frac{\lambda_i}{\mu_i} + \frac{\lambda_i^2}{2\mu_i(\mu_i - \lambda_i)} \tag{3.24}$$

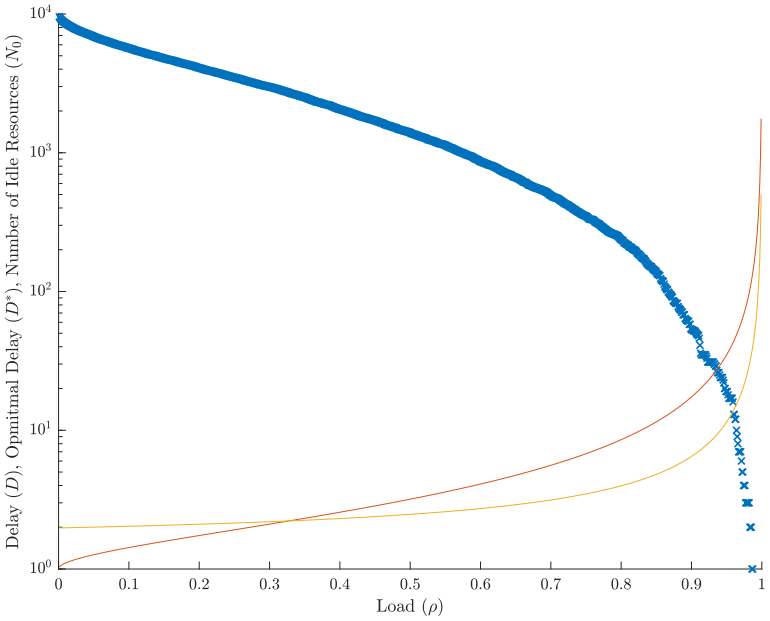

Figure 3.6: A graph depicting the delay for a system of $M/D/1$ Queues, using an Ad Hoc Solution (Yellow) and substituting in the Optimal solution for a system of $M/M/1$ queues. The Blue crosses indicate the number of Idle resources.

By observing Figure 3.6, we see that our current solution is no longer optimal. So we are now left to find the new optimal. We return to the Lagrangian equation for the non trivial solution for λ_i^*:

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \frac{1}{\lambda\mu_i}\left[1 + \frac{2\lambda_i}{2(\mu_i - \lambda_i} + \frac{2\lambda_i^2}{2(\mu_i - \lambda_i)^2}\right] - \phi_0 = 0 \tag{3.25}$$

$$\implies \frac{\lambda_i\mu_i}{(\mu_i - \lambda_i)^2} - \lambda\mu_i\phi_0 + 1 = 0 \tag{3.26}$$

This now form a quadratic equation that is easy to solve:

$$\frac{\mu_i}{\lambda\mu_i\phi_0 - 1}\lambda_i = (\mu_i - \lambda_i)^2 \tag{3.27}$$

$$\lambda_i^2 - \left(2\mu_i + \frac{\mu_i}{\lambda\mu_i\phi_0 - 1}\right)\lambda_i + \mu_i^2 = 0 \tag{3.28}$$

[1] It is worth pointing out here that for $D/D/1$ (ie both service and arrival rates are deterministic) queue, when both arrival AND Service rate are deterministic, as long as $\lambda_i < \mu_i$, there will be **No Delay** in the system at all [33]! It is the randomness in service and arrival rate that cause delays in queues.

By quadratic formula:

$$\lambda_i^o(\phi_0, \mu_i, \lambda) = \frac{\left(2\mu_i + \frac{\mu_i}{\lambda\mu_i\phi_0 - 1}\right) - \sqrt{\left(2\mu_i + \frac{\mu_i}{\lambda\mu_i\phi_0 - 1}\right)^2 - 4\mu_i^2}}{2} \tag{3.29}$$

$$= \left(\mu_i + \frac{\mu_i}{2(\lambda\mu_i\phi_0 - 1)}\right) - \frac{\mu_i}{2}\sqrt{\frac{1}{(\lambda\mu_i\phi_0 - 1)^2} + \frac{4}{\lambda\mu_i\phi_0 - 1}} \tag{3.30}$$

We are now able to see that the optimal resource allocation in this instance is:

$$\boxed{\lambda_i^* = \max\left\{0, \lambda_i^o(\phi_0, \mu_i, \lambda)\right\}} \tag{3.31}$$

Like with the M/M/1 queues, the value of ϕ_0 can be derived numerically by solving the equation:

$$\lambda = \sum_{i=1}^{N} \lambda_i$$

Thus we can now state:

Theorem 3.1. *For a system of $M/D/1$ queues, the optimal resource allocation is:*

$$\lambda_i^* = \max\left\{0, \left(\mu_i + \frac{\mu_i}{2(\lambda\mu_i\phi_0 - 1)}\right) - \frac{\mu_i}{2}\sqrt{\frac{1}{(\lambda\mu_i\phi_0 - 1)^2} + \frac{4}{\lambda\mu_i\phi_0 - 1}}\right\} \tag{3.32}$$

This gives us an optimal delay of:

$$D_{Opt} = \frac{1}{\lambda}\sum_{i=1}^{N} \frac{\lambda_i^*}{\mu_i} + \frac{\lambda_i^{*2}}{2(\mu_i - \lambda_i^*)} \tag{3.33}$$

3.6 The $M/G/1$ queue

Now we move onto an overall generalisation of the whole process. The $M/G/1$ queue looks at queues with a generalised distribution for the service rate of the servers.

$$D_i = \mathbb{E}[\mu_i^{-1}] + \frac{\lambda_i\mu_i\mathbb{E}[\mu_i^{-2}]}{2(\mu_i - \lambda_i)} \tag{3.34}$$

Thus our new objective function is:

$$D = \frac{1}{\lambda}\sum_{i=1}^{N} \lambda_i\mathbb{E}[\mu_i^{-1}] + \frac{\lambda_i^2\mu_i\mathbb{E}[\mu_i^{-2}]}{2(\mu_i - \lambda_i)} \tag{3.35}$$

The Lagrangian to solve to optimise this system is given by:

$$\mathcal{L}(\lambda_i, \phi_i, \phi_0) = D - \phi_0\left(\lambda - \sum_i \lambda_i\right) - \sum_i \phi_i\lambda_i \tag{3.36}$$

The first desired first order condition to solve it is:

$$\frac{\partial\mathcal{L}}{\partial\lambda_i} = \frac{1}{\lambda}\left[\mathbb{E}[\mu_i^{-1}] + \frac{2\lambda_i\mu_i\mathbb{E}[\mu_i^{-2}]}{2(\mu_i - \lambda_i)} + \frac{2\lambda_i^2\mu_i\mathbb{E}[\mu_i^{-2}]}{2(\mu_i - \lambda_i)^2}\right] - \phi_0 = 0 \tag{3.37}$$

$$\implies \frac{\lambda_i\mu_i^2\mathbb{E}[\mu_i^{-2}]}{(\mu_i - \lambda_i)^2} + \mathbb{E}[\mu_i^{-1}] - \lambda\phi_0 = 0 \tag{3.38}$$

This can be re arranged into a quadratic in respect to λ_i:

$$\frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]}\lambda_i = (\mu_i - \lambda_i)^2 \tag{3.39}$$

$$\lambda_i^2 - \left(\frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]} + 2\mu_i\right)\lambda_i + \mu_i^2 = 0 \tag{3.40}$$

By the Quadratic formula we now get:

$$\lambda_i^o(\phi_0, \mu_i, \lambda) = \frac{\left(\frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]} + 2\mu_i\right) - \sqrt{\left(\frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]} + 2\mu_i\right)^2 - 4\mu_i^2}}{2} \tag{3.41}$$

$$= \frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{2(\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}])} + \mu_i - \frac{\mu_i}{2}\sqrt{\left(\frac{\mu_i\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]}\right)^2 + \frac{4\mu_i\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]}} \tag{3.42}$$

Thus we now have a fully generalised formula for the system presented in [4]:

Theorem 3.2. *For a system of $M/G/1$ queues, the optimal resource allocation is:*

$$\lambda_i^* = \max\left\{0, \frac{\mu_i^2\mathbb{E}[\mu_i^{-2}]}{2(\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}])} + \mu_i - \frac{\mu_i}{2}\sqrt{\left(\frac{\mu_i\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]}\right)^2 + \frac{4\mu_i\mathbb{E}[\mu_i^{-2}]}{\lambda\phi_0 - \mathbb{E}[\mu_i^{-1}]}}\right\} \tag{3.43}$$

This gives us an optimal delay of:

$$D_{Opt} = \frac{1}{\lambda}\sum_{i=1}^{N}\lambda_i^*\mathbb{E}[\mu_i^{-1}] + \frac{\lambda_i^{*2}\mathbb{E}[\mu_i^{-2}]}{2(1-\rho_i^*)} \tag{3.44}$$

3.7 Where to go next?

So far we have managed to fully generalise the system presented in [4] to a wider class of queuing functions thanks to the most recent result regarding a system of $M/G/1$ queues shown in Theorem 3.2. We have also shown that the methodology can be used when it comes to minimising cost functions for these resource allocation problems. This leads onto the Economic Extension of this problem, where now that we have optimised the allocation of our resources, we wish to optimise the processors at our disposal to process such resources.

Chapter 4

Optimal Investment Strategies

Having investigated the idea of resource allocation in such systems, it is time consider the demand side of the problem- how do we effectively choose the servers we have in the system? In order to do so, initially it is worth looking more into the extension proposed at the end of [4] and look to build upon this further.

4.1 An Economic Problem

Suppose we have obtained an optimised delay/cost function, $D(\lambda, \boldsymbol{\lambda}^*|\boldsymbol{\mu})$, where λ is a given load, $\boldsymbol{\mu}$ is given set of resources we have at out disposal and $\boldsymbol{\lambda}^*$ is our optimal allocation of resources across the system. Now assume the load follows a distribution $p(\lambda)$. This is the pdf of the anticipated loads over a lifetime of investments. The resources we have are all of a specific type $\alpha = 1, 2, ..., A$, where $\mu_i = \mu_\alpha$ is resource i is of type α. As a result we can use this to calculate the expected profits from our investments:

$$P(\boldsymbol{\mu}) = \int d\lambda p(\lambda)[r\lambda - D(\lambda, \boldsymbol{\lambda}^*|\boldsymbol{\mu})] \tag{4.1}$$

where, r is the given *Revenue Rate* from our investment. For simplicity, we shall assume this rate is constant, which we shall discuss in more detail later on. The aim of this problem is to maximise the revenue we generate by optimising the resources we use. This will have to be in respect to Budgetary Constraint:

$$\sum_i L(\mu_i) \leq B \tag{4.2}$$

where $L(\mu_i)$ is the cost to use each resource μ_i and B is the total budget we have available, which we can not exceed. Within this setup, we can assess the following:

- What is the optimal allocation of μ_i's we can use in order to maximise our Revenue?
- What is the minimum budget required in order to setup such a business?
- What is the maximum load we can allow into the system to maximise our profits?

Let the probability distribution follow a Gamma Distribution, as this allows us to extend this result to a wide family of distributions by adjusting the parameters:

$$p_{\alpha,\lambda_0}(\lambda) = \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha \Gamma(\alpha)} e^{-\frac{\lambda}{\lambda_0}}$$

As a result this shall lead to the usage of **Incomplete Gamma Functions** and their properties:

$$\Gamma(s, x) = \int_x^\infty t^{s-1}\, \mathrm{e}^{-t}\, \mathrm{d}t \tag{4.3}$$

$$\gamma(s, x) = \int_0^x t^{s-1}\, \mathrm{e}^{-t}\, \mathrm{d}t \tag{4.4}$$

Since we previously used numerical results in order to obtain the values of our Lagrange multiplier previously, we won't always be able to rely on evaluating integral analytically. Instead we will have to use Numerical methods. Specifically, we shall be relying on Gauss Lauguerre and Lengedre Integration methods[34]. Whilst they will only provide exact results for Lauguerre and Legendre Quadratic equations, they provide a good enough approximation to use in our results. They involve evaluating the integral with the use of an appropriate weight function, $\omega(x)$, and rearrange the integral in such a way that it can be approximated by:

$$\int_a^b \omega(x)\, f(x)\, dx \approx \sum_i \omega(x_i)\, f(x_i) \tag{4.5}$$

4.1.1 Using $D_{\mathbf{OPT}}$ as a cost function

Our first step to solving this problem is to evaluate the above expression for the revenue we are expected to generate. We shall initially be using the total delay across the system as our cost function:

$$D(\phi_0, \lambda^*|\mu) = \sum_{i=1}^{N} \frac{1}{\mu_i - \lambda_i^*} = \sum_{i=1}^{N} (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.6}$$

Not is this is slightly different to the delay function form [4], since we are no longer looking at *average* delay across the system, but more so the *total* delay. We evaluate the integral as follows:

$$P(\boldsymbol{\mu}) = \int d\lambda p(\lambda) \Big[r\lambda - D(\phi_0, \lambda^*|\mu) \Big] \tag{4.7}$$

$$= \mathbb{E}\Big[r\lambda - D(\phi_0, \lambda^*|\mu) \Big] \tag{4.8}$$

$$= r\mathbb{E}[\lambda] - \mathbb{E}\Big[D(\phi_0, \lambda^*|\mu) \Big] \tag{4.9}$$

$$= r\mathbb{E}[\lambda] - \mathbb{E}\Big[\sum_{i=1}^{N} (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \Big] \tag{4.10}$$

$$= r\mathbb{E}[\lambda] - \sum_{i=1}^{N} \int d\lambda p(\lambda) (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.11}$$

With the usage of the Gamma Distribution and a *Laguerre Polynomial Integral*. We are able to evaluate the integral as follows:

$$\int d\lambda p(\lambda) (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) = \int d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^{\alpha} \Gamma(\alpha)} e^{-\frac{\lambda}{\lambda_0}} (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.12}$$

$$= \frac{1}{\lambda_0} \int d\lambda \frac{\left(\frac{\lambda}{\lambda_0}\right)^{\alpha-1} e^{-\frac{\lambda}{\lambda_0}}}{\Gamma(\alpha)} (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.13}$$

Here we let our weight function be

$$\omega(x) = \frac{x^{\alpha} e^{-x}}{\Gamma(\alpha + 1)}$$

Thus, by Gauss-Laguerre Quadrature, we can say that:

$$\frac{1}{\lambda_0} \int d\lambda \frac{\left(\frac{\lambda}{\lambda_0}\right)^{\alpha-1} e^{-\frac{\lambda}{\lambda_0}}}{\Gamma(\alpha)} (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) = \int dx \omega(x) (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.14}$$

$$\approx \sum_{x} \omega(x) (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1) \tag{4.15}$$

This is a function which can hence obtain numerically. Our expected revenue from this system is:

$$\boxed{P(\boldsymbol{\mu}) = r\lambda_0 \alpha - \sum_{i=1}^{N} \sum_{x} \omega(x) (\sqrt{\phi_0 \mu_i} - 1)\Theta(\sqrt{\phi_0 \mu_i} - 1)}$$

The next aim from here would be to maximise this Expected Revenue in respect to the budget constraint. However, before going any further, I think it is worth questioning whether this Optimal Delay function is a suitable cost function. Whilst it takes into the account the costs created by 'inefficiency' in the system, it fails to take into account the wasted cost on idle resources, the fact more powerful resources cost more to operate and so on. So for now it is best to take a step back and formulate a more realistic cost function.

4.2 Deriving a Cost Function

When we evaluated this Revenue function above, we directly used the Optimal Delay function as a proposed Cost Function. However, as previously discussed, this is not the most realistic function to use. It may make sense to have a cost function of the form:

$$C_i(\lambda, \lambda_i, \mu_i) = f_0(\mu_i) + f_1(\rho_i) \tag{4.16}$$

Here, $f_0(\mu_i)$ is the cost incurred by the delays in the system and would like it to be convex in order to get a unique and optimal solution. To account for more powerful units costing more to operate, we want $f_0(\mu_i)$ to be increasing in respect to μ_i (ie $\frac{df_0}{d\mu_i} > 0$). When we have $f_1(\rho_i)$ is the cost incurred by the processing power of the system such that $f(\rho_i) \to 0$ as $\lambda_i \to 0$ and it is inversely proportional to μ_i- a more powerful server incurs a lower cost per computational load. We shall compare these results with the onto using the resource allocation a proposed in [4] and how this affects the profits of a firm. The cost function for the whole system is:

$$C(\lambda, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{i=1}^{N} C_i(\lambda) \tag{4.17}$$

$$= \sum_{i=1}^{N} f_0(\mu_i) + f_1(\rho_i) \tag{4.18}$$

Note that $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_N)$ denotes the vector representing the allocation of resources across the system (It is not the same λ, which is jsut the total load in the system). Let's also assume that the system is made in such a way that it can only deal with a set maximum load, λ_{Max}. The system can make a fixed amount of revenue off any load below this margin, but can't process any excess, thus incurring some sort of penalty cost on these lost profits. The full profit function for the system is then given as:

$$P(\boldsymbol{\mu}) = \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[r\lambda - \sum_{i=1}^{N} f_1(\rho_i) \right] - \sum_{i=1}^{N} f_0(\mu_i) - \int_{\lambda_{\text{Max}}}^{\infty} d\lambda p(\lambda) P\left[\lambda - \lambda_{\text{Max}}\right] \tag{4.19}$$

Here, P denotes the penalty cost of being unable to utilise the full load. Notice how the $f_0(\mu_i)$ is outside of the integral. This is because it is independent of the load received. Let $C_{\text{Run}} = \sum_{i=1}^{N} f_0(\mu_i)$ dente the full baseline costs of the operation. In general:

$$P(\boldsymbol{\mu}) = \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[r\lambda - \sum_{i=1}^{N} f_1(\rho_i) \right] - \sum_{i=1}^{N} f_0(\mu_i) - \int_{\lambda_{\text{Max}}}^{\infty} d\lambda p(\lambda) P\left[\lambda - \lambda_{\text{Max}}\right] \tag{4.20}$$

$$= \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[r\lambda - \sum_{i=1}^{N} f_1(\rho_i) \right] - P\left[\int_0^{\infty} d\lambda p(\lambda) \left[\lambda - \lambda_{\text{Max}}\right] - \int_0^{\lambda_{\text{Max}}} d(\lambda) \left[\lambda - \lambda_{\text{Max}}\right] \right] - C_{\text{Run}} \tag{4.21}$$

$$= \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[r\lambda - \sum_{i=1}^{N} f_1(\rho_i) + P[\lambda - \lambda_{\text{Max}}] \right] - P\mathbb{E}\left[\lambda - \lambda_{\text{Max}}\right] - C_{\text{Run}} \tag{4.22}$$

$$= \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[(r+P)\lambda - \left(\sum_{i=1}^{N} f_1(\rho_i) + \lambda_{\text{Max}} \right) \right] - P\left(\lambda_0 \alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}} \tag{4.23}$$

$$= \int_0^{\lambda_{\text{Max}}} d\lambda \frac{e^{-\frac{\lambda}{\lambda_0}}}{\Gamma(\alpha)} \left[(r+P) \left(\frac{\lambda}{\lambda_0}\right)^{\alpha} - \frac{\lambda_{\text{Max}}}{\lambda_0} \left(\frac{\lambda}{\lambda_0}\right)^{\alpha-1} - \sum_{i=1}^{N} f_1(\rho_i) \right] - P\left(\lambda_0 \alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}} \tag{4.24}$$

$$= \frac{1}{\Gamma(\alpha)} \left[\lambda_0 (r+P) \gamma(\alpha+1, \lambda_{\text{Max}}) - \lambda_{\text{Max}} \gamma(\alpha, \lambda_{\text{Max}}) \right] - C_{\text{Run}} \tag{4.25}$$

$$- \sum_{i=1}^{N} \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i) - P\left(\lambda_0 \alpha - \lambda_{\text{Max}}\right) \tag{4.26}$$

Note that once the integral $\int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i)$ is evaluated analytically[1], it will be in the form:

$$\int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i) = \frac{1}{\Gamma(\alpha)} g_i(\mu_i; \boldsymbol{\theta}) \gamma(h_i(\alpha), \lambda_{\text{Max}}) \tag{4.27}$$

where $\boldsymbol{\theta}$ represents the parameters of $f_1(\rho_i)$,$g_i(\mu_i; \boldsymbol{\theta})$ is the function derived form the evaluated integral and $h_i(\alpha)$ determines the shape of the associated incomplete Gamma function. Thus we can express our profits as:

$$\text{P}(\boldsymbol{\mu}) = \frac{1}{\Gamma(\alpha)} \left[\lambda_0 (r+P) \gamma(\alpha+1, \lambda_{\text{Max}}) - \lambda_{\text{Max}} \gamma(\alpha, \lambda_{\text{Max}}) - \sum_{i=1}^{N} g_i(\mu_i; \boldsymbol{\theta}) \gamma(h_i(\alpha), \lambda_{\text{Max}}) \right] - P\left(\lambda_0 \alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}} \tag{4.28}$$

In this form it is easier to see how profits behave in respect to the various parameters at play and allows the firm to make changes such as to maximise their profits. All operating and penalty costs as well as the distribution of incoming load may be out of their control, so they are unable to adjust for these. However the revenue rate, r, the Maximum Load in the System, λ_{Max} and the servers at their disposal, μ_i, is well in their control. It is easy to see that $P(\boldsymbol{\mu})$ is linear in respect to r, so the obvious choice there is to set it as high as possible, which of course depends heavily upon market limitations. However, it is the allocation of servers and maximum load are harder to distinguish, and we shall go into more detail. But first I think it is worth evaluating this equation with potential candidates for $f_0(\mu_i)$ and $f_1(\rho_i)$. To simplify notation, we shall use the variables P_{FULL} and R, where:

$$P_{\text{FULL}} = P\left(\lambda_0 \alpha - \lambda_{\text{Max}}\right) \tag{4.29}$$

$$R = \lambda_0 (r+P) \gamma(\alpha+1, \lambda_{\text{Max}}) - \lambda_{\text{Max}} \gamma(\alpha, \lambda_{\text{Max}}) \tag{4.30}$$

These values are unaffected by the type of cost function we use. We shall also be evaluating our functions with the Ad-Hoc Allocation $\lambda_i = \frac{\lambda}{\mu} \mu_i$.

4.2.1 Fully Linear Cost Function

For $f_0(\mu_i)$, we shall initially be use $A\mu_i$. As for $f_1(\rho_i)$, we shall simply let it equal to $\zeta \frac{\lambda_i}{\mu_i}$, where ζ is a scaling factor of the cost depending upon the load of the server. Firstly, it is simple to see that:

$$C_{\text{Run}} = A \sum_{i=1}^{N} \mu_i \tag{4.31}$$

$$\int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i) = \zeta \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \frac{\lambda_i}{\mu_i} \tag{4.32}$$

We then evaluate $f_1(\rho_i)$ alters with the Ad-Hoc Allocation, $\lambda_i^{\text{Ad-Hoc}} = \frac{\lambda}{\mu} \mu_i$:

$$\zeta \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \frac{\lambda_i^{\text{Ad-Hoc}}}{\mu_i} = \frac{\zeta}{\mu} \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \lambda \tag{4.33}$$

$$= \frac{\zeta}{\mu \Gamma(\alpha)} \int_0^{\lambda_{\text{Max}}} d\lambda \left(\frac{\lambda}{\lambda_0} \right)^{\alpha} e^{-\frac{\lambda}{\lambda_0}} \tag{4.34}$$

$$= \frac{\zeta \lambda_0}{\mu \Gamma(\alpha)} \gamma\left(\alpha+1, \lambda_{\text{Max}}\right) \tag{4.35}$$

Thus our overall profit is given by:

$$P(\boldsymbol{\mu}) = \frac{1}{\Gamma(\alpha)} \left[R - \frac{N \zeta \lambda_0}{\mu} \gamma\left(\alpha+1, \lambda_{\text{Max}}\right) \right] - P_{\text{FULL}} - A \sum_{i=1}^{N} \mu_i \tag{4.36}$$

[1] We will notice later on that it may not be able to always be evaluated analytically and numerical solutions with no functional form will be required.

4.2.2 Fully Quadratic Cost Function

We shall now test it out for Quadratic Functions. So let $f_0(\mu_i) = A_0 + A_1\mu_i + A_2\mu_i^2$ and $f_1(\rho_i) = \zeta_0 + \zeta_1\rho_i + \zeta_2\rho_i^2$. Only $f_1(\rho_i)$ alters as we alter the resource allocation:

$$f_1(\rho_i^{\text{Ad-Hoc}}) = \zeta_0 + \zeta_1\frac{\lambda}{\mu} + \zeta_2\left(\frac{\lambda}{\mu}\right)^2 \tag{4.37}$$

Using these functions, we are able to evaluate the expected revenue under these circumstances.

$$C_{\text{Run}} = \sum_{i=1}^{N} A_0 + A_1\mu_i + A_2\mu_i^2 \tag{4.38}$$

$$= NA_0 + A_1\sum_{i=1}^{N}\mu_i + A_2\sum_{i=1}^{N}\mu_i^2 \tag{4.39}$$

$$\int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i^{\text{Ad-Hoc}}) = \int_0^{\lambda_{\text{Max}}} d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha\Gamma(\alpha)} e^{-\frac{\lambda}{\lambda_0}}\left(\zeta_0 + \zeta_1\frac{\lambda}{\mu} + \zeta_2\left(\frac{\lambda}{\mu}\right)^2\right) \tag{4.40}$$

$$= \int_0^{\lambda_{\text{Max}}} d\lambda \frac{e^{-\frac{\lambda}{\lambda_0}}}{\Gamma(\alpha)}\left(\frac{\zeta_0}{\lambda_0}\left(\frac{\lambda}{\lambda_0}\right)^{\alpha-1} + \frac{\zeta_1}{\mu}\left(\frac{\lambda}{\lambda_0}\right)^{\alpha} + \frac{\zeta_2\lambda_0}{\mu^2}\left(\frac{\lambda}{\lambda_0}\right)^{\alpha+1}\right) \tag{4.41}$$

$$= \frac{\lambda_0}{\Gamma(\alpha)}\left(\frac{\zeta_0}{\lambda_0}\gamma\left(\alpha, \lambda_{\text{Max}}\right) + \frac{\zeta_1}{\mu}\gamma\left(\alpha+1, \lambda_{\text{Max}}\right) + \frac{\zeta_2\lambda_0}{\mu^2}\gamma\left(\alpha+2, \lambda_{\text{Max}}\right)\right) \tag{4.42}$$

Thus our overall profit is given by:

$$P(\boldsymbol{\mu}) = \begin{array}{c} \frac{1}{\Gamma(\alpha)}\left[R - N\lambda_0\left(\frac{\zeta_0}{\lambda_0}\gamma\left(\alpha, \lambda_{\text{Max}}\right) + \frac{\zeta_1}{\mu}\gamma\left(\alpha+1, \lambda_{\text{Max}}\right) + \frac{\zeta_2\lambda_0}{\mu^2}\gamma\left(\alpha+2, \lambda_{\text{Max}}\right)\right)\right] \\ -P_{\text{FULL}} - \left(NA_0 + A_1\sum_{i=1}^{N}\mu_i + A_2\sum_{i=1}^{N}\mu_i^2\right) \end{array} \tag{4.43}$$

4.2.3 Fully Exponential Cost Function

We shall now test it out for Quadratic Functions. So let $f_0(\mu_i) = A_0 e^{A_1\mu_i}$ and $f_1(\rho_i) = \zeta_0 e^{\zeta_1\rho_i}$.

$$\sum_{i=1}^{N} f_0(\mu_i) = \sum_{i=1}^{N} A_0 e^{A_1\mu_i} = A_0\sum_{i=1}^{N} e^{A_1\mu_i} \tag{4.44}$$

$$\int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) f_1(\rho_i^{\text{Ad-Hoc}}) = \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda)\zeta_0 e^{\zeta_1\frac{\lambda}{\mu}} \tag{4.45}$$

$$= \frac{\zeta_0}{\Gamma(\alpha)}\int_0^{\lambda_{\text{Max}}} d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha} e^{-\frac{\lambda}{\lambda_0}} e^{\zeta_1\frac{\lambda}{\mu}} \tag{4.46}$$

$$= \frac{\zeta_0}{\Gamma(\alpha)}\int_0^{\lambda_{\text{Max}}} d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha} e^{-\lambda\left(\frac{1}{\lambda_0} - \frac{\zeta_1}{\mu}\right)} \tag{4.47}$$

Let $\widetilde{\lambda_0} = \left(\frac{1}{\lambda_0} - \frac{\zeta_1}{\mu}\right)^{-1}$. Thus:

$$\frac{\zeta_0}{\Gamma(\alpha)}\int_0^{\lambda_{\text{Max}}} d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha} e^{-\lambda\left(\frac{1}{\lambda_0} - \frac{\zeta_1}{\mu}\right)} = \frac{\zeta_0}{\Gamma(\alpha)}\int_0^{\lambda_{\text{Max}}} d\lambda \frac{\lambda^{\alpha-1}}{\lambda_0^\alpha} e^{-\frac{\lambda}{\widetilde{\lambda_0}}} \tag{4.48}$$

$$= \frac{\zeta_0\widetilde{\lambda_0}^{\alpha-1}}{\Gamma(\alpha)\lambda_0^\alpha}\int_0^{\lambda_{\text{Max}}} d\lambda \left(\frac{\lambda}{\widetilde{\lambda_0}}\right)^{\alpha-1} e^{-\frac{\lambda}{\widetilde{\lambda_0}}} \tag{4.49}$$

$$= \frac{\zeta_0\widetilde{\lambda_0}^{\alpha}}{\Gamma(\alpha)\lambda_0^\alpha}\gamma(\alpha, \lambda_{\text{Max}}) \tag{4.50}$$

$$= \frac{\zeta_0\widetilde{\lambda_0}^{\alpha}}{\Gamma(\alpha)\lambda_0^\alpha}\gamma(\alpha, \lambda_{\text{Max}}) \tag{4.51}$$

$$= \frac{\zeta_0}{\Gamma(\alpha)}\left(\frac{1}{1 + \frac{\zeta_1\lambda_0}{\mu}}\right)^{\alpha}\gamma(\alpha, \lambda_{\text{Max}}) \tag{4.52}$$

Now we can evaluate our profit function as follows:

$$P(\boldsymbol{\mu}) = \frac{1}{\Gamma(\alpha)}\left[R - N\zeta_0\left(\frac{1}{1+\frac{\zeta_1\lambda_0}{\mu}}\right)^{\alpha}\gamma(\alpha,\lambda_{\text{Max}})\right] - P_{\text{FULL}} - A_0\sum_{i=1}^{N}e^{A_1\mu_i} \tag{4.53}$$

4.2.4 Summary

We now have a family of profit functions using Ad-Hoc Allocations to optimise with the resources at our disposal:

- Linear Costs

$$P(\boldsymbol{\mu}) = \frac{1}{\Gamma(\alpha)}\left[R - \frac{N\zeta\lambda_0}{\mu}\gamma\left(\alpha+1,\lambda_{\text{Max}}\right)\right] - P_{\text{FULL}} - A\sum_{i=1}^{N}\mu_i \tag{4.54}$$

- Quadratic Costs

$$\begin{aligned} P(\boldsymbol{\mu}) = \quad & \tfrac{1}{\Gamma(\alpha)}\Big[R - N\lambda_0\left(\tfrac{\zeta_0}{\lambda_0}\gamma\left(\alpha,\lambda_{\text{Max}}\right) + \tfrac{\zeta_1}{\mu}\gamma\left(\alpha+1,\lambda_{\text{Max}}\right) + \tfrac{\zeta_2\lambda_0}{\mu^2}\gamma\left(\alpha+2,\lambda_{\text{Max}}\right)\right)\Big] \\ & -P_{\text{FULL}} - \left(NA_0 + A_1\textstyle\sum_{i=1}^{N}\mu_i + A_2\sum_{i=1}^{N}\mu_i^2\right) \end{aligned} \tag{4.55}$$

- Exponential Costs

$$P(\boldsymbol{\mu}) = \frac{1}{\Gamma(\alpha)}\left[R - N\zeta_0\left(\frac{1}{1+\frac{\zeta_1\lambda_0}{\mu}}\right)^{\alpha}\gamma(\alpha,\lambda_{\text{Max}})\right] - P_{\text{FULL}} - A_0\sum_{i=1}^{N}e^{A_1\mu_i} \tag{4.56}$$

Note, due to the modular nature of these functions, that we can mix and match these accordingly. for example a processor may have a linear f_0 but an exponential f_1, but for now we shall use this structure. So how do we go about optimising these functions? Well the main areas of interest, as mentioned previously, would be choose an appropriate allocation of resources as well as a suitable maximum load, λ_{Max}, in order to maximise our profits.

4.3 Optimising Profits

4.3.1 The relation between Maximum Load and Overall Profits

The first thing to note about our profit function is how there are several Incomplete Gamma Functions in the system, all dependent on the value of λ_{Max}. This implies very different values for the profit, depending upon the value we set for λ_{Max}, as can be seen in the figure below. Here, we see that, thanks to the Incomplete Gamma functions, the Profit exhibits a sigmoidal shape in respect to the Maximum Load the system can manage, thus creating two distinct regimes. Notice how the function is also now concave, so usual convex optimisation methods won't always work. Instead, it is good to focus on looking at these distinct regimes separately and observing what properties they may exhibit.

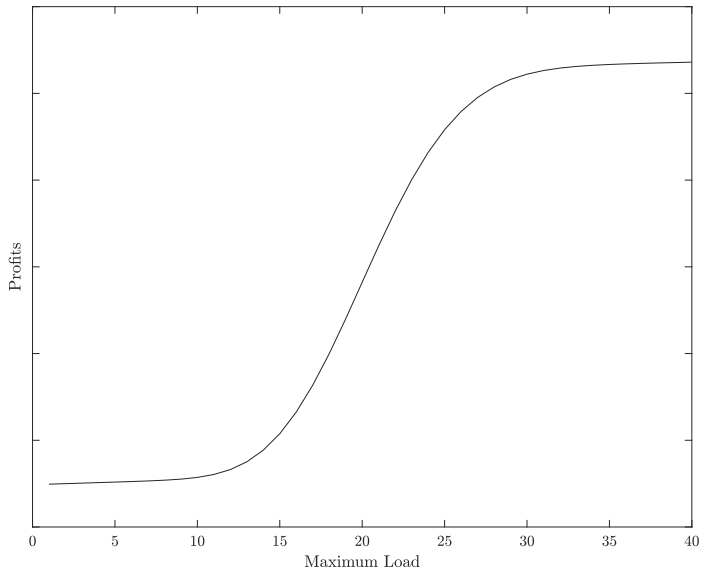

Figure 4.1: A graph exhibiting how profits change as we change the variable λ_{Max} ion a system where the load follows a Gamma Distribution, Gamma$(10, 20)$ using the Linear Profit function (4.54)

In particular, the regimes can be distinguished as follows:

- Sometimes we could have system that is unable to keep up with the demand we receive. This mean that we have very small λ_{Max} in relation to the distribution. under such circumstances, the Profit becomes:
$$P(\boldsymbol{\mu}) \to -P\left(\lambda_0\alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}} \tag{4.57}$$
By noticing the parameters in this case, it is evident that the firm will always be in negative profits, and thus always losing money. This shows that if a firm is unable to accommodate for such a capacity, they shouldn't even think of going into business, regardless of any revenue rates or costs.

- On the other hand, we can have a system, which is able to handle to load received with ease, being able to handle large capacities of load. For such cases we have a large λ_{Max},resulting in the Profits becoming:
$$P(\boldsymbol{\mu}) \to \frac{1}{\Gamma(\alpha)}\left[\lambda_0(r+P) - \lambda_{\text{Max}} - \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta})\right] - P\left(\lambda_0\alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}} \tag{4.58}$$
Here it is a lot harder to see how profitable the business, and we will have to instead take into account the Budget Constraint and cost functions.

Whilst this can be seen as a fairly trivial result, it highlights the importance in our model to accommodate for high loads in order to avoid large penalty costs. The maximum load we can process comes in part from the servers we choose to have since $\lambda_{\text{Max}} < \mu$. In order to accommodate for these large loads, we need to optimise the servers we have, by using the budget that has been set for us- which what shall focus upon next.

4.3.2 Utilising the Budget Constraint to Break Even

Now returning back to the scenario explained at the beginning of this section. Suppose, we have an overall Budget of B. This mean that in order for this business to be profitable in the first place, we must at least *break even.*:

$$P(\boldsymbol{\mu}) \geq B \tag{4.59}$$

This is when the profits produced by the business are greater than or equal to the initial budget we set out with. By using the profit-making regime, we observe that:

$$\implies \frac{1}{\Gamma(\alpha)}\left[\lambda_0(r+P) - \left(\sum_{i=1}^{N} f_0(\mu_i) - \lambda_{\text{Max}}\right) - \sum_{i=1}^{N} g(\mu_i;\boldsymbol{\theta})\right] - P\left(\lambda_0\alpha - \lambda_{\text{Max}}\right) \geq B \tag{4.60}$$

By simplifying this expression, we get the following **Break Even Condition**:

$$\boxed{\lambda_0(r+P) + \lambda_{\text{Max}} \geq B\Gamma(\alpha) + \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta}) + P\Gamma(\alpha)\left(\lambda_0\alpha - \lambda_{\text{Max}}\right) + \sum_{i=1}^{N} f_0(\mu_i)}$$

4.3.3 Optimal Resource Allocation

Now that we have established this economic model, we can now return to the central focus of this paper. So far we have just been using an Ad-Hoc allocation in proportion to the service rate of the server. But how can we optimise this in order to minimise costs? This is where we can use the results obtained in the first half of this paper, where we used resource allocation to minimise average delay times. Now we need to use it to maximise profits. To refine our approach, we firstly need to notice that the only part of the profit function dependent on the allocation of resources is the Cost Function, which we need to alter slightly:

$$C(\lambda, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{i=1}^{N} f_0(\mu_i) + f_1(\rho_i) \tag{4.61}$$

Notice that this is very similar to minimising our cost functions in the previous chapter with some minor differences:

$$\text{Minimise} \quad \frac{1}{\lambda}\sum_{i=1}^{N}\lambda_i D_i \quad \rightarrow \quad \text{Minimise} \quad \sum_{i=1}^{N} D_i \tag{4.62}$$

We no longer have the factor of $\frac{1}{\lambda}$ and each term in the sum is no longer multiplied by λ_i. By using our now optimal load $\rho_i^o = \frac{\lambda_i^o}{\mu_i}$ we can adjust the solutions properly and obtain the following set of optimal loads for each system:

- A quadratic cost function:

$$f_1(\rho_i) = d_0\rho_i + d_1\rho_i^2 \qquad \Rightarrow \qquad \rho_i^0 = \frac{1}{2d_1}(\mu_i\phi_0 - d_0)$$

- A cubic cost function:

$$f_1(\rho_i) = d_0\rho_i + d_1\rho_i^2 + d_2\rho_i^3 \qquad \Rightarrow \qquad \rho_i^0 = \frac{1}{3d_2}\Big[-d_1 + \sqrt{d_1^2 + 3d_2(\mu_i\phi_0 - d_0)}\Big]$$

- An exponential cost function. [2]:

$$f_1(\rho_i) = e^{d_0\rho_i} - 1 \qquad \Rightarrow \qquad \rho_i^0 = \frac{1}{d_0}\ln\frac{\phi_0\mu_i}{d_0}$$

- A cost function akin to the $M/M/1$ delay functions:

$$f_1(\rho_i) = d_0\frac{\rho_i}{1-\rho_i} \qquad \Rightarrow \qquad \rho_i^0 = 1 - \sqrt{\frac{d_0}{\mu_i\phi_0}}$$

[2]Because we are no longer have the factor of λ_i on the outside of the exponential, we can no actually find a functional for for the allocation

4.4 Optimal Investment Strategy

Another usage for our budget constraint is to optimise which servers we choose to have within our system. Different servers have different costs to purchase, often depending on their service rate. This budget is expressed:

$$\sum_i L(\mu_i) \leq B$$

This can be use as a possible Kuhn-Tucker condition to optimise the profits by. However, before we do that, it is worth noting that since we are optimising in respect to $\boldsymbol{\mu}$:

$$\frac{1}{\Gamma(\alpha)}\left[\lambda_0(r+P)\gamma(\alpha+1,\lambda_{\text{Max}}) - \lambda_{\text{Max}}\gamma(\alpha,\lambda_{\text{Max}}) - \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta})\gamma(h_i(\alpha),\lambda_{\text{Max}})\right] - P\left(\lambda_0\alpha - \lambda_{\text{Max}}\right) - C_{\text{Run}}$$

$$\propto -\left[C_{\text{Run}} + \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta})\gamma(h_i(\alpha),\lambda_{\text{Max}})\right]$$

Now let:

$$\mathcal{C}(\boldsymbol{\mu}) = C_{\text{Run}} + \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta})\gamma(h_i(\alpha),\lambda_{\text{Max}}) \tag{4.63}$$

it is easy to see that maximising $P(\boldsymbol{\mu})$ is equivalent to minimising $\mathcal{C}(\boldsymbol{\mu})$ in respect to $\boldsymbol{\mu}$.

Thus our problem is equivalent to:

Maximise	Minimise
$P(\boldsymbol{\mu})$	$\mathcal{C}(\boldsymbol{\mu})$
with respect to:	with respect to:
$\sum_i L(\mu_i) \leq B$	$\sum_i L(\mu_i) \leq B$
$\mu_i > 0 \quad i = 1, 2, .., N$	$\mu_i > 0 \quad i = 1, 2, .., N$

We have simply transformed this from a Profit Maximisation Problem to its equivalent Cost Minimisation Problem.[3]. We are equipped with technique from before to solve this problem. Our Lagrangian is of the form:

$$\mathcal{L}(\boldsymbol{\mu},\psi_B,m_i) = \mathcal{C}(\boldsymbol{\mu}) - \psi_B\left[\sum_i L(\mu_i) - B\right] - \sum_{i=1}^{N} m_i\mu_i \tag{4.64}$$

$$= \sum_{i=1}^{N} f_0(\mu_i) + \sum_{i=1}^{N} g_i(\mu_i;\boldsymbol{\theta})\gamma(h_i(\alpha),\lambda_{\text{Max}}) - \psi_B\left[\sum_i L(\mu_i) - B\right] - \sum_{i=1}^{N} m_i\mu_i \tag{4.65}$$

By differentiating in respect to μ_i, we get the following KKT Conditions:

i We are using our full budget

$$\frac{\partial\mathcal{L}}{\partial\mu_i} = \frac{\partial f_0}{\partial\mu_i}(\mu_i) + \frac{\partial g}{\partial\mu_i}(\mu_i;\boldsymbol{\theta})\gamma(h(\alpha),\lambda_{\text{Max}}) - \psi_B\frac{\partial L}{\partial\mu_i}(\mu_i) = 0 \tag{4.66}$$

ii We are not using all of our budget

$$\frac{\partial\mathcal{L}}{\partial\mu_i} = \frac{\partial f_0}{\partial\mu_i}(\mu_i) + \frac{\partial g}{\partial\mu_i}(\mu_i;\boldsymbol{\theta})\gamma(h(\alpha),\lambda_{\text{Max}})) = 0 \tag{4.67}$$

In order to solve this we need to pick suitable functions for $f_0(\mu_i)$, $f_1(\rho_i)$ and $L(\mu_i)$. Like before, we shall be looking at Linear, Quadratic and Exponential cases:

- **Linear Costs:** Firstly lets assume the cost of a unit i is merely an scaled up factor in respect to its computational power. Thus let $L(\mu_i) = L\mu_i$. By using linear f_0 and f_1, we can evaluate the above conditions as:

$$\frac{\partial\mathcal{L}}{\partial\mu_i} = A - \psi_B L = 0 \tag{4.68}$$

[3]This is a basic concept in Economics and the equivalence should be fairly trivial

It is obvious to see that having such structure for the function won't work in this cases as all μ_i's disappear in in the first order conditions, so it is best to trial with other functions instead.

- **Quadratic Costs:** Now, lets assume the cost of a unit i is Quadratic in respect to its computational power. Thus let $L(\mu_i) = L_0 + L_1\mu_i + L_2\mu_i$. By using quadratic f_0 and f_1, we can evaluate the above conditions as:

$$\frac{\partial \mathcal{L}}{\partial \mu_i} = (A_1 + 2A_2\mu_i) - \psi_B(L_1 + 2L_2\mu_i) \tag{4.69}$$

$$= A_1 - \psi_B L_1 + 2(A_2 + \psi_B L_2)\mu_i = 0 \tag{4.70}$$

$$\implies \mu_i^* = \max\left\{\frac{-A_1 + \psi_B L_1}{2(A_2 + \psi_B L_2)}, \frac{-A_1}{2A_2}\right\} \tag{4.71}$$

- **Exponential Costs:** Now, lets assume the cost of a unit i is Quadratic in respect to its computational power. Thus let $L(\mu_i) = L_0 e^{L_1\mu_i}$. By using quadratic f_0 and f_1, we can evaluate the above conditions as:

$$\frac{\partial \mathcal{L}}{\partial \mu_i} = A_0 A_1 e^{A_1\mu_i} - \psi_B L_0 L_1 e^{L_1\mu_i} \tag{4.72}$$

$$= A_1 - \psi_B L_1 + 2(A_2 + \psi_B L_2)\mu_i = 0 \tag{4.73}$$

$$\implies \mu_i^* = \max\left\{\frac{-A_1 + \psi_B L_1}{2(A_2 + \psi_B L_2)}, \frac{-A_1}{2A_2}\right\} \tag{4.74}$$

This gives us an optimal μ_i we should set across a homogeneous system. We find that the value of μ_i is homogeneous too across the system, which makes sense for a set of evenly distributed resources as with this case. However, whilst this equation has given us a solution, it hasn't actually answer the question we set out to answer. It has given us the optimal power to set to our resources if we were able to produce them in a bespoke manner in order to maximise our profits. In reality, though, this isn't necessarily how it works. These resources will often come in fixed packages and we can't freely choose a specific power. Instead we have to make do with what is available. In order to do so, we can no longer rely on our previous methods, due the now discrete nature of this problem. Instead, we will need to reformulate certain aspects of this problem and provide a new methodology- as we shall do in the next chapter.

Chapter 5

Extending our problem to a discrete setting

5.1 Model Setup

So far, we have been assuming that our load and service rate is infinitely divisible and continuous. However in reality this isn't necessary the case. For example, if we are simply serving people in a queue, the only come in units of full person- Not a fraction of a person! Likewise with service rates, when it comes to something like a computer processor, they only come in fixed sizes. Due to this, it is now worth reformulating our problem, but this time in a discrete setting. Firstly we establish that both μ_i and λ_i come in pre-set packages such that:[1]

$$\mu_i \in \tilde{\boldsymbol{\mu}} = \{\mu^A, \mu^B, ..\} \quad \forall i = 1, 2, .., S$$
$$\lambda_j \in \tilde{\boldsymbol{\lambda}} = \{\lambda^A, \lambda^B, ..\} \quad \forall j = 1, 2, .., P$$

where S and P denote the number of different types of packets arriving and types of servers available respectively. We also now need to redefine the definitions of λ and μ:

$$\lambda = \sum_{\alpha} M_\alpha \lambda^\alpha$$
$$\mu = \sum_{\beta} K_\beta \mu^\beta$$

where M_α denotes the number of packets of type α being used and K_β denotes the number of servers of type β in the system. Let the cost of a to run a processor of type i be:

$$L(\mu^i) = L^i \quad \forall i = 1, 2, .., S$$

Let the revenue generated by a package of type j be:

$$R(\lambda^j) = r^j \quad \forall j = 1, 2, .., P$$

We can now reformulate our two main problems as follows:

- The problem from [4] now becomes:

  $$\text{Minimise } D = \frac{1}{\lambda} \sum_{i=1}^{N} \frac{\lambda_i}{\mu_i - \lambda_i}$$

  in respect to the constraints:

  $$\lambda = \sum_{\alpha} M_\alpha \lambda^\alpha$$
  $$\lambda_i \in \tilde{\boldsymbol{\lambda}} \quad i = 1, 2, .., N$$

- The Economic Problem now becomes about:

  $$\text{Maximise } P(\boldsymbol{\mu}) = \sum_{\alpha} p(\lambda)[r^\alpha M_\alpha - C(\lambda, \boldsymbol{\lambda}^* | \boldsymbol{\mu})] \tag{5.1}$$

  In Respect to the Budget Constraint:

  $$\sum_{\beta} K_\alpha L^\beta \leq B$$

To solve such problems, we now need to use a mixture of our discrete and continuous optimisation methods. First, we shall look to solve the Optimal Routing Problem in this discrete setting. Firstly by setting the servers to be discrete then see what happens when both our incoming packages and incumbent servers are discrete.

[1] We shall be using subscripts to denote which unit we are referring to and superscripts for the type of unit it may be

5.1.1 Optimal Routing: Discrete Servers

Firstly lets assume that our servers come in fixed sizes. For example, they could grow in an exponential manner

$$\mu_i \in A_1 = \{0.1, 1, 10, 100\}$$

or can just be uniformly distributed over a range:

$$\mu_i \in A_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Just introducing a set of discrete servers doesn't require too much computation analytically, since the methodology in [4] still works as we still optimise in respect to a continuous variable. What is of interest is seeing how the number of idle resources differs in this case. We simulated this process on several sets for μ and can bee seen below. One thing to notice is how the idle resource curve now is of the form of a step function. What each step indicates is which type of servers are active an not. For example, look at the first graph. The first step represents the number of servers which are below $\mu_i = 10$, the next step then excludes those with $\mu_i = 9$, and this trend continues till about the halfway load (Which is when just have servers with $\mu_i = 1$ idle) and then it drops off as all workers are now working. Whilst they may just seem trivial, it emphasises how our system now primarily prioritises high production servers with work as opposed to low production servers. This is even more noticeable when the servers differ in larger scales of magnitude as shown in the second graph. Here We have idle resources until we are very close to a full load. With the few powerful resources doing most the work, the majority of servers in the system are mostly idle.

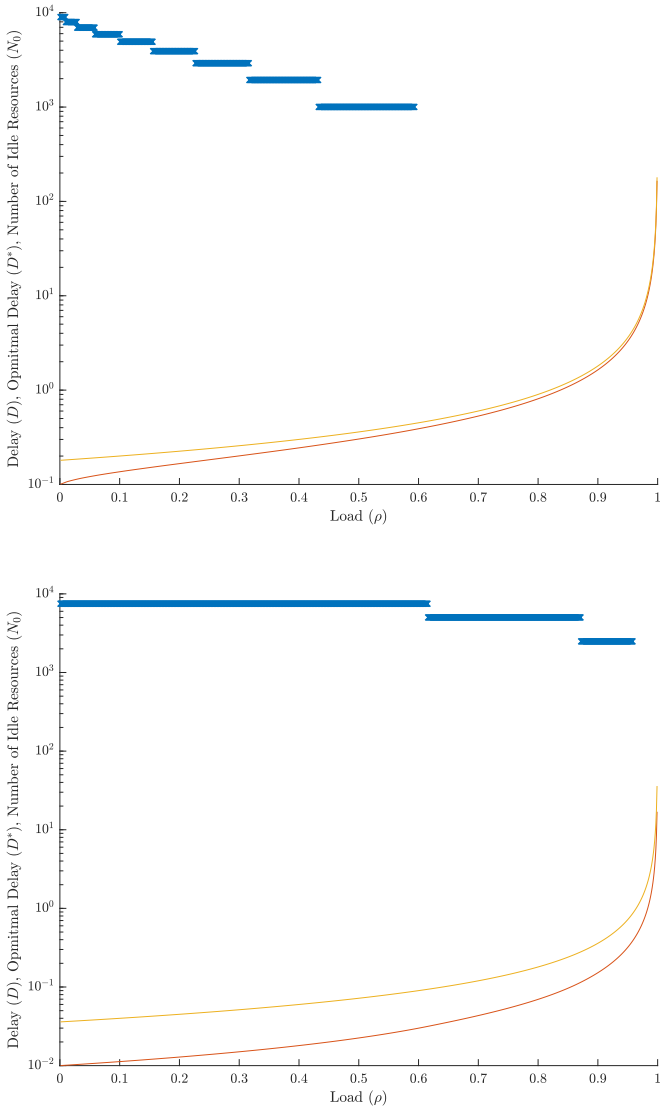

Figure 5.1: These graphs depict the the allocation using servers from the set A_1 (Top) and from the set A_2

5.1.2 Optimal Routing: A Fully Discrete System

Now that have completed the optimisation in respect to discrete servers, we can now observe what happens when we make the resources discrete too. We have a set load and total service rate for the system:

$$\lambda = \sum_{\alpha} M_\alpha \lambda^\alpha$$

$$\mu = \sum_{\beta} K_\beta \mu^\beta$$

As part of the resource allocation problem we want to do is optimise our M_α for the whole system. For example, in the case of having servers belonging to the set $\tilde{\boldsymbol{\lambda}} = \{1, 10, 100, 1000\}$, we shall obtain a random sample of $\{M_1, M_{10}, M_{100}, M_{1000}\}$.

The values of each M_α is extracted from a suitable probability distribution. One such distribution would be a geometric distribution of the form:

$$\forall \alpha \in \tilde{\boldsymbol{\lambda}} : \quad \mathbb{P}(m) = (1 - r_\alpha) r_\alpha^m \qquad (r_\alpha < 1)$$

Where $\mathbb{P}(m)$ is the probability there exists m servers of type α. One thing to notice about the parameter, r_α, is the following:

$$\lambda^{a_1} < \lambda^{a_2} < \lambda^{a_3} \quad \Longrightarrow \quad r_{a_1} > r_{a_2} > r_{a_3}$$

This implies that the probability decays exponentially as we look at larger load sizes. We are then able to use this density to obtain the *expected number*, $\mathbb{E}(m_\alpha)$, of each type of server:

$$\mathbb{E}(m_\alpha) = \sum_{\alpha} (1 - r_\alpha) r_\alpha^m m_\alpha \tag{5.2}$$

$$= (1 - r_\alpha) \underbrace{x \frac{d}{dx} \sum_{m} (r_\alpha x)^m}_{\sum_m m(x^m r_\alpha)\Big|_{x=1}} \tag{5.3}$$

$$= (1 - r_\alpha) \sum_{m} m r_\alpha^m \tag{5.4}$$

This provides us with the following generating function:

$$G_\alpha(x) = (1 - r_\alpha) \sum_{m} r_\alpha^n x^m \tag{5.5}$$

using the properties of generating functions, we know that:

$$\begin{aligned}
\mathbb{E}(m_\alpha) &= x G_\alpha^{'}(x)\Big|_{x=1} \\
&= (1 - r_\alpha) \frac{\partial}{\partial x} \left\{ \frac{1}{1 - r_\alpha x} \right\}\Big|_{x=1} \\
&= \frac{r_\alpha}{1 - r_\alpha}
\end{aligned}$$

As $r_\alpha \to 1$, the mean diverges. This essentially shows that we expect a large number of packets of a small size and a small number of packages of a larger size.

Now that we have our distribution of loads, we can choose an appropriate sample:

$$\lambda_{\text{Samp}} = \sum_{\alpha} m_\alpha \lambda_\alpha$$

This ties nicely into the investment problem too. suppose we want to minimise the costs, $f_1(\rho_i)$, across the system, where ρ_i is the usual load each server receives. Let $M_{\alpha,i}$ denote the number of packets of size α being assigner to server i. Thus we can say that for each server:

$$\lambda_i = \sum_{\alpha} M_{\alpha,i}\lambda^\alpha \quad \Longrightarrow \quad \rho_i = \frac{\sum_\alpha M_{\alpha,i}\lambda^\alpha}{\mu_i}$$

This ensures we get a fair allocation of resources we get the following optimisation problem to solve:

Minimise:

$$\sum_i f(\rho_i)$$

in respect to:

$$\lambda_i = \sum_{\alpha} M_{\alpha,i}\lambda^\alpha \leq \mu_i$$

$$\lambda = \sum_{\alpha} M_\alpha \lambda^\alpha$$

Here we wish to optimise the value of $M_{\alpha,i}$ for each server. However, this develops into a very complex combinatorics problem which can't be solved with ease. However this leaves us with a valid formulation left for us to solve with an appropriate method.

5.2 Optimal Investments with Optimised Allocation

Suppose we have system which is required to produce a fixed power, μ, to process the entire load. This power must greater than or equal to λ_{Max}. We have at our disposal a set processors:

$$\tilde{\boldsymbol{\mu}} = \{\mu^A, \mu^B, ..., \mu^B\}$$

We wish to optimise the number of each server to have in our system. In order to so, we have developed the following simple algorithm that we shall explain in this section:

Algorithm 1 Optimal Configuration Algorithm

```
1:  begin
2:  Initialise: {μ^A, μ^B, ..}, μ = λ_max, Conf
3:  for a = 0 : ⌊μ/μ^A⌋ do
4:      for b = 0 : ⌊μ/μ^B⌋ do
5:          ...
6:          if aμ^A + bμ^B + ... = μ then
7:              config = [a, b, ...]
8:              Conf = Conf ∪ config
9:          end
10:     ...
11: end
12: Initialise: {P^A, P^B, ..}, μ, Feas, InFeas
13: for C ∈ Config do
14:     Costs = Budget − Σ_α C_α P^α
15:     if Costs > 0 then
16:         Feas = Feas ∪ C
17:     else:  InFeas = InFeas ∪ C
18:     end
19: end
20: Initialise: Pr(x), Optimum Profit = 0
21: for C ∈ Feas do
22:     if Pr(C) > Optimum Profit then
23:         Optimum Profit = Pr(C)
24:         Optimum Configuration = C
25:     end
26: end
27: return: Optimum Configuration, Optimum Profit
28: end
```

How it works:

- **Combinatorics (Lines 2 to 11):** In this section of the algorithm by using the required power output and the set of resources we can get, we perform a grid search to obtain all combinations of these servers which give us the required power for the system. This gives us a feasible configuration set which narrows down our search space for the remainder of the process.
- **Budgetary Constraints (Lines 12 to 19):** In this section we further narrow down the search space by eliminating all combinations which don't satisfy our budgetary constraints and place them into a Feasible Set.
- **Evaluating Profits (Lines 20 to 26):** With our remaining combinations, we compute the profits obtained from each combination.
- **Optimal Output (Line 27):** We now sort our final set in order to obtain the combination of servers which produces the optimal server allocation.

Upon completion of this algorithm, we apply the previously established optimal resource allocations to the system and as a result we we obtain and optimum state for the system to operate in order to minimise our costs as well as maximise profits.

5.2.1 A Numerical Example

For a demonstration of this algorithm at work, we shall use some numerical results shown below:

- We have servers which can be chose from the set :

$$\texttt{Servers} = \{1, 10, 100, 1000\}$$

- Now lets set our maximum demand as $\mu = \lambda_{\max}$. The maximum load we can process from the system won't exceed the total service rate, so we use this as our upper bound. We let the Maximum Load be $\lambda_{\max} = 1000$ and initiate the first stage of the algorithm which shall return all combinations of servers which can at least satisfy this requirement. By doing so- we reduce our search space from infinite to just 562 possible combinations.

$$\texttt{Combinations = Size of Conf = 562}$$

- We now look to implement our budget constraint into this solution , By setting our budget to $B = 3,000$, we work out the cost of the remaining units and see if they fit within this budget. We end up finding that only 307 of these are within out budget constraint.

$$\texttt{Affordable Combinations = Size of Feas = 307}$$

- Now we have our refined search space, we can find the optimal allocation by finding what combinations provide us with the best profit. We substitute this into the Profit function and ad on the remainder of the budget we had. We then sort to find the combination with the largest profit and the algorithm returns:

$$\texttt{Optimal Allocation= } \{450, 50, 50, 0\}$$

Now that an optimal server investment has been made, all that is left is to allocate the resources in an efficient manner, in order to obtain optimsed profits.

Chapter 6

Discussion and Conclusion

As we come to the end of this paper, it is worth summarising the key results we have obtained and also look at any possible shortfalls of these solutions and where we can go from here.

6.1 Resource Allocation

In regards to the problem of Optimal Resource Allocation, we observed how the distribution of idle servers at a given load differs as we alter the distribution of the servers across the system, noticing how when we have a set of steep exponentially distributed servers, we have more idle servers at low loads, whereas when we have a more flat distribution of servers, the number of idle servers drops off pretty quickly as the load is increased.

We then generalised the framework for this problem so we could use it to minimise different types of functions. This included introducing a upper bound constraint for cost functions that didn't diverge at maximum load. We produced results for both quadratic and exponential functions, as well as more traditional delay functions for the $M/D/1$ and $M/G/1$ queues. Due to the direct relation between the $M/G/1$ queue and the $M/M/1$ and $M/D/1$, the equivalence between these delay functions can be easily shown.

Proposition 6.1. *The optimal delay time for a system of $M/D/1$ or $M/M/1$ can be directly derived from the delay for a system of $M/G/1$ queues*

Proof. The best way to show this is to show that both $M/M/1$ and $M/D/1$ formulae can be derived from the $M/G/1$ formula.

$$D_i = \mathbb{E}[\mu_i^{-1}] + \frac{\lambda_i \mathbb{E}[\mu_i^{-2}]}{2(1-\rho_i)}$$

- For $M/M/1$ queue, the servers serve at an exponential rate. Thus, $\mathbb{E}(\mu_i^{-1}) = \mu_i^{-1}$ and $\mathbb{E}(\mu_i^{-2}) = 2\mu_i^{-2}$

$$D_i = \mu_i^{-1} + \frac{\lambda_i \mu_i^{-2}}{1-\rho_i} = \frac{\mu_i^{-1}}{1-\rho_i} = \frac{1}{\mu_i - \lambda_i} \tag{6.1}$$

- For $M/D/1$ queue, the servers serve at an exponential rate. Thus, $\mathbb{E}(\mu_i^{-1}) = \mu_i^{-1}$ and $\mathbb{E}(\mu_i^{-2}) = \mu_i^{-2}$

$$D_i = \mu_i^{-1} + \frac{\lambda_i \mu_i^{-2}}{2(1-\rho_i)} = \mu_i^{-1} + \frac{\lambda_i}{2\mu_i(\mu_i - \lambda_i)} \tag{6.2}$$

□

Whilst we now have these neat usable analytical solutions, whilst investigating other types of queues, the results aren't as simple. For example, consider the case of the $M/M/1/\theta$ queue, an $M/M/1$ queue with a finite capacity of θ in each queue. This has a delay function[17]:

$$D_i = \frac{1}{\lambda_i}\Big[\frac{1}{1-\rho_i} + \frac{1}{1-\rho_i^{\theta-1}} - \frac{\theta\rho_i^{\theta+1}}{1-\rho_i^{\theta-1}}\Big] \tag{6.3}$$

This will produced a polynomial to solves, which is only solvable analytically for capacities of 1, 2 (which leaves a cubic equation to solve with the Cardano Method), 3 and 4. For any larger capacities, it becomes impossible to obtain any analytical solutions and only numerical solutions can be obtained. This shows that whilst we have a robust method transferable to many other functions, it doesn't come without its limitations.

6.2 Investment Strategies and the Economic Problem

The remainder of this paper then focused upon solving an economic problem derived from solving the other side of the Resource Allocation Problem. For this problem we produced a profit function to measure the profits gained from the system, trying out several functional forms and seeing how these could be optimised. We observed the different ways in which we could optimise λ_{Max}, μ_i and the resource allocation across the system done in such a way to minimise *Total* Costs.

6.2.1 Refining our Profit Function

For our Economic Problem, we were using a Profit function of the form:

$$P(\boldsymbol{\mu}) = \int_0^{\lambda_{\text{Max}}} d\lambda p(\lambda) \left[r\lambda - \sum_{i=1}^{N} f_1(\rho_i) \right] - \sum_{i=1}^{N} f_0(\mu_i) - \int_{\lambda_{\text{Max}}}^{\infty} d\lambda p(\lambda) P\left[\lambda - \lambda_{\text{Max}}\right] \tag{6.4}$$

Whilst it provides a decent evaluation within our framework there are aspects which could be developed further:

- **Constant Revenue Rate (r):** We have assumed that revenue rate we obtain from our output is constant. However, as previously mentioned this is no necessarily the best way to model the revenue. For example, we are usually serving a market with with a limited number of customers, as well with external competition. These factors imply we would be experiencing diminishing returns on the more we sell-this is whee the marginal output of a process *decreases* as another factor involved within the production *increases*[35] . In order to solve this problem, instead of having r, we should have some revenue function $r(\lambda; \boldsymbol{\theta})$- where we have load dependent revenue rate for the system that takes into account the diminishing returns principle in respect to the total load through the system. Expanding further, if we were to make this a multiple firm problem, where let's say we have a market of F firms, where firm j revenue function of $r_j(\lambda_j; \boldsymbol{\theta}_j)$, then we will also have to take into account some sort of interacting terms with other firms across the industry. Whilst this is very computationally heavy and requires a lot more work behind it, it takes us a step closer to having a much more realistic model we can apply to real world affairs.

- **Expected Profits:** Upon evaluation, this function provides us with the Expected profits, which is dependent upon the distribution the load into the system follows. However, this doesn't necessarily provide a full picture into the reliability of the process. For example, in our methodology, we assumed that $\lambda \sim \text{Gamma}(\alpha, \lambda_0)$. The expected value is given by $\mathbb{E}(\lambda) = \alpha\lambda_0$. This means that there will be several combinations of the parameters of the distribution that will have the same mean. Thus we are obtaining a mean but missing the shape of the distribution in the output. This can create some issues as we essentially have the knowledge of the amount of risk in the process removed. The mean may be found at a sharp peak, thus implying a large amount of profits but also high risk. Thus even when profits are evaluated, you also need to bear in mind. the distribution of the load.

- **Heterogeneous Cost Functions:** We are assuming the cost to run all processors follows the exact same method, but this may not necessarily hold. For example, suppose we have two sets of processors, one being a very powerful one whereas the other one is old a weak. The more powerful processor's costs or processing a load may just grow linearly in increasing power. However, for the weaker set, this growth in cost could be exponential in respect to the load. This is realistic to assume just by observing the nature of the evolution of computers over the past few decades. Instead, to make the whole process a lot more realistic, it may be best to used heterogeneous cost functions instead. In particular, we can make the $f_1(\rho_i)$'s in the system become $f_{1,i}(\rho_i)$, where the functional form remains the sames, we have a difference in parameters from unit to unit. This adjusts for the scaling differences between large and small resources.

6.2.2 Dealing with ϕ_0

For the majority of the Economic Problem, we were dealing with mainly analytical results that can be visualised numerically. However, the optimal solutions could only obtained numerically. This is due to the nature of the Lagrange multiplier ϕ_0. This is because it is both a function of the total load, λ, and set of servers, $\boldsymbol{\mu}$, but can't be expressed explicitly. This raises issues when integrating over λ when evaluating out profits and differentiating by μ_i when optimising our resources. We briefly touched upon dealing with the first problem, which can be overcome with the help of numerical integration. However, with the second issue, we can resolve this without having to rely on numerics.

In order to obtain ϕ_0 in the first place, we used the identity:

$$\lambda = \sum_{i=1}^{N} \lambda_i^*$$

where our optimal allocation is of the form $\lambda_i^* = \min\{\max\{\lambda_i^o, 0\}, 1\}$. Note that our optimum allocation, λ_i^o, is in fact a function entirely dependent upon dependent on the servers, total resource and ϕ_0:

$$\lambda_i^o = \lambda_i^o(\phi_0, \mu_i, \lambda)$$

Thus, we define the following function:

$$g(\lambda, \boldsymbol{\mu}, \phi_0) = \sum_i \lambda_i^* - \lambda \equiv 0 \tag{6.5}$$

We shall focus upon the $f_1(\rho_i)$ part of the profit function, as this is the part affected by resource allocation. Let the total of these costs be denoted by:

$$C_1(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_i \int d\lambda p(\lambda) f_1(\rho_i^*) \tag{6.6}$$

By differentiating in respect to μ_i we get the following:

$$\frac{\partial C_1}{\partial \mu_i} = \int_0^{\lambda_{\max}} \rho(\lambda) \left[f_1'(\rho_i^*(\lambda)) \frac{\partial \rho_i^*(\lambda)}{\partial \mu_i} + \sum_j f_1'(\rho_j^*(\lambda)) \frac{\partial \rho_j^*(\lambda)}{\partial \phi_0} \frac{\partial \phi_0}{\partial \mu_i} \right] \tag{6.7}$$

Note that we have to use the product rule here since not only is it a function of each server, μ_i, but also on ϕ_0, which is dependent on **all** servers, causing there to be a set of coupled equations. The derivative $\partial\phi_0/\partial\mu_i$ needed to evaluate Equation 6.7, and it is obtained from our function $g(\lambda, \boldsymbol{\mu}, \phi_0)$, that fixes the Lagrange parameter ϕ_0, giving the following identity:

$$\frac{\partial \phi_0}{\partial \mu_i} = \frac{\frac{\partial g}{\partial \mu_i}}{\frac{\partial g}{\partial \phi_0}} .$$

Whilst this successfully gives us a solution for $\frac{\partial \phi_0}{\partial \mu_i}$, when considering the rest of our Lagrangian equation, we will actually have to go through the motions to determine ψ_B in case the budget constraint becomes an active constraint, because the actual value of ψ_B needed to keep the budget constraint will directly influence the optimal strengths μ_i^* of the resources of various types i.

6.2.3 Bridging the Gap with Economics

When it comes to the bigger picture, whilst the contents of this paper have mainly looked at extending the mathematics problem into an Economic Problem. However, we haven't gone into much detail how this links into wider Economics and its literature. Specifically, this is essentially an Industrial Economics Problem. If this research is to be extended, a link with Industrial Economics will be key, and hopefully an overlap in results will be attained. In addition to this, Industrial Economics focuses upon processes in wide scale industries and how companies seek to maximise profits according to the environment they find themselves in with fellow competitors. This adds another layer to our problem as we now have to consider competition to the firm we are already modelling for.

6.3 Optimising our Investments

The methodology proposed at the end of Section 4.4, provided us with a way to choose optimal investments when we have bespoke resources, which we could alter to our own specifications, at our disposal. Whilst this isn't the most accessible solution in most industries, we have shown an option is there if it is feasible. However if we are being bespoke, we should also be more specific with the kind of functions we are using. In our solution we strictly used the same functional form for the fixed cost, running cost and pricing of all our resources. In reality these won't follow the same form. Load dependent running costs tend to be exponential whereas the cost to buy a certain processor often follows a tiered approach, where processors with a power within a fixed range will cost the same price. The price of a unit depends on which predetermined pool its provider feels it fits into. This results in possible step functions a discontinuities within the graph. This makes usage of calculus non trivial and might be unsolvable in such a way. This then results us again taking a step into the direction of discrete optimisation to adjust for this.

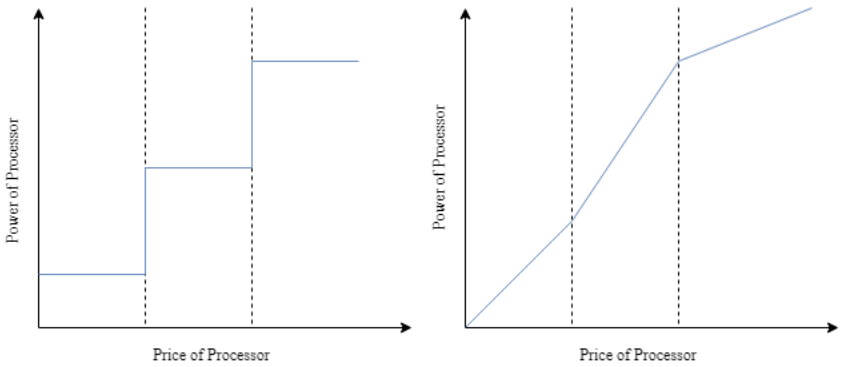

Figure 6.1: A depiction of how a tiered pricing system would work. The first graph shows the same when the price is constant across a tier. The second graph shows a tiered price system, where the price of a processor is determined by the function abided by the tier it falls within.

6.3.1 Refining our algorithm

At the end of our final chapter, an algorithm is proposed which helps us find the optimal allocation of servers for our system. It does this by steadily reducing our search space step by step, before running through all remaining combinations in order to obtain the optimum. Whilst we have show it works and can be used to get optimal results, for much larger systems we will run into issues regarding running time since there can be vast numbers of different combinations we can have. Whilst not enough time was available to perform this, a future extension would be to refine this and make it a lot more efficient so it runs better on large systems. there are a variety of tried and tested methods which could be used to help the process, such as simulated annealing or even evolutionary algorithms like Differential Evolution. In addition to this, further aspects of the algorithm composition can be looked into. For example, at the moment, it doesn't consider the money left over from the excess budget from the purchase of resources. By combining refinements to the economics problem with improvements to the algorithm, will only improve our final result. We have talked a lot regarding optimal resource allocation and optimal investments, but to gain the most out of them, we need more of a combination between the two, which has been mostly lacking up till now. This is the pickup point for any further developments with this research.

6.4 Final Remarks

Now that we have discussed potential shortfalls and what the next steps to take from here in regards to our research, we shall one final question a sceptical reader might have had from the outset: Why are we buying resources that we don't plan on using in the first place? To some it may seem reasonable to cut ones' losses and not waste money on such a purchase. The answer lies within the finer details of the question. It is not that we are *never* using these resources. Rather, it is that we *rarely* use these resources. Let's consider the scenario depicted in Figure 6.2. We have a set of servers that come in five fixed sizes. After allocating resources efficiently, we observe the now usual phenomenon of there existing idle servers for low loads in the system, where the weakest type of resource isn't used until we reach a load of 0.9. However, what is the probability of being at any of the stages of idle servers? This will usually be determined by the probability distribution of λ, $p(\lambda)$. So far we have assumed that we have a high probability for lower loads and a low probability for high loads.

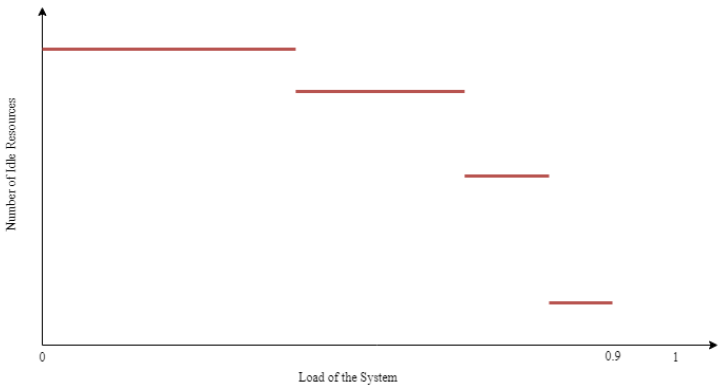

Figure 6.2: A graph depicting the number of idle servers in a large scale process and how this changes with load of the system

Let's assume that we remove all servers that remained idle till $\rho = 0.9$. We are left with fewer servers and a reduction in the total power the system can provide. This new power is given by:

$$\tilde{\mu} = \mu - \Delta$$

This causes a rescaling in the load with $\rho = \frac{\lambda}{\mu}$ being replaced with $\tilde{\rho} = \frac{\lambda}{\tilde{\mu}}$. After simulating the process again with this reused set of servers we obtain Figure 6.3. We now have no idle servers beyond a load of 0.6. Whilst this solution may seem a more efficient allocation of resource, all is not as it seems. first arrives the issue of rescaling. The x-axis is now in terms of $\tilde{\rho}$ instead of ρ. Thus skewing perceptions of the likelihood of these events. I fact, the likelihood of having no idle processors is identical in both scenarios, since the probability depends solely on λ- not μ. Instead, what we have done, by cutting down on the processing power, is reduce the maximum load, λ_{Max}, we are able to process. All this achieves is a reduction in Revenue and increase of incurred Penalty Costs- thus we may just end up paying up more than the low investment required for a few low power servers. This all goes to show that in many cases, all is not as it seems and it is worth investigating deeper.

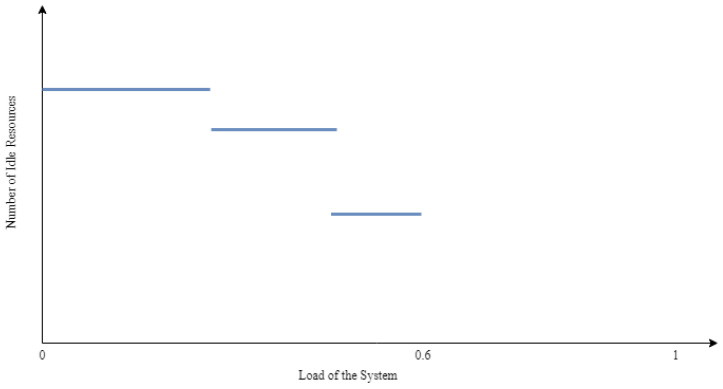

Figure 6.3: A graph depicting the number of idle servers in the same process as Figure 6.2 but this time with the least powerful set of servers

Appendix
MATLAB Codes used

- Code used to derive results for an M/M/1 Queue

```
clear
clc
N=10000;
mui = rand(1,N);
%%%LOOP FOR CHANGING DISTRIBUTION OF mui%%%
%SET = [1,2,5,10];
%for p = SET;

%% NORMAL %%
%mui = normrnd(100,10*p,[1,N]);
%for r = 1:N;
%while (mui(r)<0)
%    mui(r) = normrnd(100,10*p);
%end
%end

%% Exponential %%
%mui = exprnd(p,[1,N]);

%% Gamma %%
%mu0=p;
%Beta=0.5;
%mui = gamrnd(mu0,Beta,[1,N]);
%%
mu = sum(mui);
 for k = 1:1000
lambda = mu*k/1000;
LOAD(k)=k/1000;
    beta0 = fzero(@(beta) Sfun(beta,mui,mu,lambda), [0, 1000]);
    phi0 = 1/(lambda*beta0);
    % check that the solution works with the orignal equation
    chek = sum( ...
        sqrt(mui/(lambda*phi0)).*heaviside(mui-sqrt(mui/(lambda*phi0))) ...
        + mui.*heaviside(sqrt(mui/(lambda*phi0))-mui)) - (mu - lambda);
    max(abs(chek));  % should be small
for j = 1:N;
l(j)= (mui(j)*lambda/mu);
D(j)=l(j)/(mui(j)-l(j));
I(j) = sum(heaviside(-1+sqrt(1/(mui(j)*lambda*phi0))));
end
DOPT(k)= sum( ...
    (sqrt(mui*lambda*phi0)-1).*heaviside(sqrt(mui*lambda*phi0)-1))/lambda;
P(k)=phi0;
DELAY(k)= sum(D)/lambda;
IDLE(k)=sum(I);
 end
 %% Graphing %%
scatter(LOAD,IDLE,'x')
hold on
plot(LOAD,DOPT,'-','HandleVisibility','off')
hold on
plot(LOAD,DELAY,'HandleVisibility','off')
xlabel('Load ($\rho$)','Interpreter','latex')
ylabel('Delay ($D$), Opmitmal Delay ($D^*$), Number of Idle Resources ...
    ($N_0$)','Interpreter','latex')
%end
set(gca,'TickLabelInterpreter','latex')
set(gca, 'YScale', 'log');

%% Function to obtain phi0 %%
function S = Sfun(beta,mui,mu,lambda)
  S = sum( sqrt(beta*mui).*heaviside(mui-beta) ...
      + mui.*heaviside(beta-mui)) - (mu - lambda);
end
```

- Code used to obtain Numerical results for Quadratic cases

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   otimize resource allocation for quadratic cost-function
%   z == Phi_0 Lagrange parameter controling sum of loads
%   Determine lambda as function of z =lammbda*phi0 !!!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global N M0 b c la z z0  mui mu
%-------------------------------------------------------------------------
%  initialization
%-------------------------------------------------------------------------
N  = 10000;              % number of processes
M0 =    1;      % exponentially distibituted resources MEAN
a  =    1;      % parameters of cost function
b  =    1;
c  =    1;
mui = rand(1,N);

SET = [1,2,5,10];
for p = SET;
mu0=1;
Beta=0.5*p;
mui = gamrnd(mu0,Beta,[1,N]);       % exprnd(M0,1,N);   % resource parameters
        % total capacity
mui = exprnd(p,[1,N]);
mu  = sum(mui);
%-------------------------------------------------------------------------
mdat= [];
zi= 0.0001:0.01:50;
Nl=length(zi);


for i=1:Nl
  z =zi(i);
  la = f(z);
  x_i = la/mu;
  D = N*(x_i.*(1+ 0.5*b*x_i +c*x_i.^2/3))/la;
  if ((la > 0) & (la < mu))
    phi0 = z/la;
    x0i = max(4*c*(z*mui-1),0);
    x0i = min((sqrt(b^2 +x0i) -b)/(2*c),1);
    D_OPT = sum(x0i.*(1+ 0.5*b*x0i +c*x0i.^2/3))/la;

    mdat = [mdat; la/mu, phi0, sum((x0i <0 )),sum((x0i >= 1)),D_OPT,D,D-D_OPT];
  end
end
scatter(mdat(:,1),mdat(:,3),'x')
hold on
plot(mdat(:,1),mdat(:,4),'o')
hold on
plot(mdat(:,1),mdat(:,5))
hold on
plot(mdat(:,1),mdat(:,6))
%plot(mdat(:,1),mdat(:,2))
end
set(gca, 'YScale', 'log');
xlim([0,1])
xlabel('Load ($\rho$)','Interpreter','latex')
%ylabel(' Langrange Parameter ($\phi_0$)','Interpreter','latex')
ylabel('Delay ($D$), Opmitmal Delay ($D^*$), No. of Idle ($N_0$) and Full ...
     ($N_F$) Resources ','Interpreter','latex')
set(gca,'TickLabelInterpreter','latex')
function la = f (z)
global b c mu mui
  x0 = max(4*c*(mui*z-1),0);
  x0 = min((sqrt(b^2 +x0) -b)/(2*c),1);
  la = sum(mui.*x0);
end
```

- Code to obtain Profits

```
%% Generating Servers%%
N=1000;
mui = rand(1,N);
for p=1:N
mui2(p)=mui(p)^2;
end
mu=sum(mui);
%% Obtaining Phi0%%
 for k = 1:1000
lambda = mu*k/1000;
beta0 = fzero(@(beta) Sfun(beta,mui,mu,lambda), [0 200]);
phi0 = 1/(lambda*beta0);
chek = sum( sqrt(mui/(lambda*phi0)).*heaviside(mui-sqrt(mui/(lambda*phi0))) ...
        + mui.*heaviside(sqrt(mui/(lambda*phi0))-mui)) - (mu - lambda);
max(abs(chek));  % should be small
Phi(k)=phi0;
for j = 1:N
    phimu(k,j)=sqrt(mui(j)*phi0);
end
LOAD(k)= lambda;
 end
for i=1:40
%% Distribution Parameters %%
l0=1000;
alpha=2;
%% Revenue Parameters %%
l_max=i;
P=2;
%% Cost Function Parameters %%
zeta=1;
Zeta0=.2;
Zeta1=.2;
Zeta2=.2;
A=5;
A0=1;
A1=1;
A2=1;
%% Loss Function %%
L=1;
L0=1;
L1=1;
L2=1;
%% Full Penalty%%
P_FULL= P*(l0*alpha - l_max);
%% Full Revenue%%10
R = l0*(r+P)*gammainc(alpha+1,l_max);
%% Expected Profits %%
Lin_PFT(i) = ...
    (R-(A*mu-l_max)*gammainc(alpha,l_max))-((N*zeta*l0)/mu)*gammainc(alpha+1,l_max)-P_FULL;
Quad_PFT(i) = (R-(N*A0+A1*mu+A2*sum(mui2)-l_max)*gammainc(alpha,l_max) - ...
    N*l0*(Zeta0*gammainc(alpha,l_max)/l0...
    + Zeta1*gammainc(alpha+1,l_max)/mu + ...
        Zeta2*l0*gammainc(alpha+2,l_max)/(mu^2)))- P_FULL;
%Exp_PFT(i) = ...
    (R-(A0*exmu-l_max)*gammainc(alpha,l_max))-N*Zeta0(1/(1+(Zeta1*l0)/mu))^alpha*gammainc(alpha,l_
%% Optimal Profits %%
%P(i) = r*l0*alpha - ...
    sum(integral(sqrt(mui*phi0)-1).*heaviside(sqrt(mui*phi0)-1),-Inf,Inf);
X_VAL(i)= l_max;
end
%% Graphic Reuslts%%
plot(X_VAL,Lin_PFT)
hold on
plot (X_VAL,Quad_PFT)
%set(gca, 'YScale', 'log');
xlabel('Maximum Load','Interpreter','latex')
ylabel('Profits','Interpreter','latex')
set(gca,'TickLabelInterpreter','latex')
function S = Sfun(beta,mui,mu,lambda)
  S = sum( sqrt(beta*mui).*heaviside(mui-beta) ...
      + mui.*heaviside(beta-mui)) - (mu - lambda);
end
```

- Code to obtain Server Combinations

```
clc, clear all
%% Running Costs %%
A0=1;
A1=1;
A2=1;
Z0=1;
Z1=1;
Z2=1;
%% Distribution Parameters %%
l_bar=20;
l0=l_bar;
alpha=20;
%% Revenue Parameters %%
l_max=5;
P=2;
r=10;
%% Combinatorics %%
U=10;
U1=U^(0);
U2=U;
U3=U^2;
U4=U^3;
mu=1000;
Conf=[];
for i=0:floor(mu/U1)
    for j= 0:floor(mu/U2)
        for k=0:floor(mu/U3)
            for l=0:floor(mu/U4)
                if i*U1+U2*j+U3*k+U4*l==mu; %Checking if combination is feasible
                    Config = [i,j,k,l];
                    N= i+j+k+l;
                    Conf=[Conf;Config N];
                else ;
                end

            end
        end
    end
end
Power= [Conf(:,1)*U1 Conf(:,2)*U2 Conf(:,3)*U3 Conf(:,4)*U4];
%% Generating Server Vector %%
for i = 1:length(Conf)
    Vec = {U1*ones(1,Conf(i,1)) U2*ones(1,Conf(i,2)) U3*ones(1,Conf(i,3)) ...
        U4*ones(1,Conf(i,4))};
    mui(i,:)= Vec;
end
%% Budgeting %%
Budget = 3000;
P=5;
P1=P;
P2=P^2;
P3=P^3;
P4=P^4;
Feas=[];
InFeas=[];
for i=1:length(Conf)
    Costs  =Budget-( P1*Conf(i,1) + P2*Conf(i,2)+P3*Conf(i,3)+P4*Conf(i,4));
    if ge(Costs, 0);
        Feas = [Feas; Conf(i,:) Costs];
    else InFeas = [InFeas; Conf(i,:) Costs];
    end
end
Feas;
InFeas;
%Conf = [Conf Costs'];
%Conf = sortrows(Conf,5);
%% Fixed Values %%
P_Full= P*(l0*alpha - l_max)
R=l0*(r+P)*gammainc(alpha+1,l_max)-l_max*gammainc(alpha,l_max)
%% Ad-Hoc Profit %%
Prof=[];
for i = 1:length(Feas)
Lin_Prof(i)= R- Feas(i,5)*Z0*l0/mu*gammainc(alpha+1,l_max)-P_Full-A0*mu;
Quad_Prof(i)= ...
    R-Feas(i,5)*l0*(Z0*l0*gammainc(alpha,l_max)+Z1/mu*gammainc(alpha+1,l_max)...
```

```
74  +Z2*l0/(mu^2)*gammainc(alpha+2,l_max)) -P_Full - ...
        (Feas(i,5)*A0+A1*sum(Feas(i,1:4))+A2*sumsqr(Feas(i,1:4)));
75  Prof=[Prof; Lin_Prof(i) Quad_Prof(i)];
76  end
77  Full = [Feas Prof];
78  Full=sortrows(Full,8);
79  OptAll=Full(1,1:4)
```

Bibliography

[1] Shu Liang, Xianlin Zeng, Guanpu Chen, and Yiguang Hong. Distributed sub-optimal resource allocation via a projected form of singular perturbation. *arXiv preprint arXiv:1906.03628*, 2019.

[2] Costis Maglaras, John Yao, and Assaf Zeevi. Optimal Price and Delay Differentiation in Queueing Systems. *SSRN Electronic Journal*, aug 2013.

[3] Shoshana Anily and Moshe Haviv. Line Balancing in Parallel M/M/1 Lines and Loss Systems as Cooperative Games. *Production and Operations Management*, 26(8):1568–1584, aug 2017.

[4] R. Kühn and S. Mostafa Mostafavi. Optimal routing policy. *IEEE Communications Letters*, 12:222–224, 2008.

[5] Transport for London — Every Journey Matters. Bus fleet data audits, Mar 2019.

[6] Peter White. Public transport: privatization and investment. *Transport Policy*, 1(3):184 – 194, 1994.

[7] Jacob Loveless, Sasha Stoikov, and Rolf Waeber. Online algorithms in high-frequency trading. *Queue*, 11(8):30:30–30:41, August 2013.

[8] Richard C Larson and Amedeo R Odoni. *Urban operations research*. Number Monograph. 1981.

[9] JFC Kingman. The first erlang century—and the next. *Queueing Systems*, 63(1-4):3, 2009.

[10] Carlos Hernández-Suárez, Carlos Castillo-Chavez, Osval López, and Karla Hernández-Cuevas. An application of queuing theory to SIS and SEIS epidemic models. *Mathematical Biosciences and Engineering*, 7(4):809–823, oct 2010.

[11] Edward H. Kaplan, David L. Craft, and Lawrence M. Wein. Analyzing bioterror response logistics: the case of smallpox. *Mathematical biosciences*, 185 1:33–72, 2003.

[12] François Baccelli and Serguei Foss. On the saturation rule for the stability of queues. *Journal of Applied Probability*, 32(2):494–507, 1995.

[13] John D. C. Little. OR FORUM—Little's Law as Viewed on Its 50th Anniversary. *Operations Research*, 59(3):536–549, jun 2011.

[14] John D. C. Little. A Proof for the Queuing Formula: L = λ W . *Operations Research*, 9(3):383–387, nov 2008.

[15] Felix Pollaczek. Über eine aufgabe der wahrscheinlichkeitstheorie. i. *Mathematische Zeitschrift*, 32(1):64–100, 1930.

[16] A Ya Khintchine. Mathematical theory of stationary queues. *Matem. Sbornik*, 39:73–84, 1932.

[17] David G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *Ann. Math. Statist.*, 24(3):338–354, 09 1953.

[18] Chih Ping Li and Michael J. Neely. Delay and rate-optimal control in a multi-class priority queue with adjustable service rates. In *Proceedings - IEEE INFOCOM*, pages 2976–2980, 2012.

[19] Laurence D Hoffman and Gerald L Bradley. *Calculus for business, economics, and the social and life sciences*. McGraw-Hill, 2010.

[20] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.

[21] Tinne Kjeldsen. A contextualized historical analysis of the kuhnâtucker theorem in nonlinear programming: The impact of world war ii. *Historia Mathematica*, 27:331–361, 11 2000.

[22] Alpha C Chiang. Fundamental methods of mathematical economics. 1984.

[23] Jacky Cresson and Frédéric Pierret. Continuous versus discrete structures I – Discrete embeddings and ordinary differential equations. nov 2014.

[24] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002.

[25] L.M. Rere, Mohamad Ivan Fanany, and Aniati Arymurthy. Simulated annealing algorithm for deep learning. volume 72, 11 2015.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[27] H Edwin Romeijn and Robert L Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5(2):101–126, 1994.

[28] Chih-ping Li and Michael J. Neely. Delay and Power-Optimal Control in Multi-Class Queueing Systems. jan 2011.

[29] W Whitt. On approximations for queues, iii: Mixtures of exponential distributions. *AT&T*

Bell Laboratories Technical Journal, 63(1):163–175, 1984.

[30] Harry J Holzer, Lawrence F Katz, and Alan B Krueger. Job queues and wages. *The Quarterly Journal of Economics*, 106(3):739–768, 1991.

[31] Shinji Ito and Ryohei Fujimaki. Large-scale price optimization via network flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3855–3863. Curran Associates, Inc., 2016.

[32] Chen Chen and Lee Kong Tiong. Using queuing theory and simulated annealing to design the facility layout in an agv-based modular manufacturing system. *International Journal of Production Research*, 57(17):5538–5555, 2019.

[33] FL Mannering, WP Kilareski, and SS Washburn. Principles of highway engineering and traffic analysis, (draft), 2003.

[34] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1965.

[35] William D Nordhaus. Geography and macroeconomics: New data and new findings. *Proceedings of the National Academy of Sciences*, 103(10):3510–3517, 2006.

Self Reflection and Assessment

Personal Thoughts

In regards to how I found this project, I found it quite an enjoyable yet challenging experience. Due to the nature of the topic, it was far removed from any of the content covered in any of the material in the Masters Programme. Thus, it required a lot of external to get my head round certain aspects of the project. It was a Mathematical Modelling problem which only really one paper of substance to base it off. Luckily, I had taken courses in Operational Research and Optimisation Theory in my undergraduate degree and these helped me tremendously. At times when speaking with my supervisor regarding the topic, I felt I knew more than the supervisor in certain, which was in a way a good and bad thing. Whilst at times it would give me confidence in my own ability, other times it was difficult to convince them at times to use alternative methods to what we would initially be suggesting. But this helped me develop my skills at communicating my academic knowledge in an effective manner. Having said that though, my supervisor was great and I really couldn't ask for much more. They were always willing to help out whenever I had questions. Even though they were poor with email communications, if I swung by his office he was very cooperative. There were a few weeks in the summer where I ended up popping in every day. They had a clear initial plan for where the project should head, but we ended up taking a few detours and looking at some alternative problems, after pointing out some flaws in the way they were thinking about the problem. In fact we did a lot more that what I could cover in this report. But in order to maintain a linear narrative throughout, I had to choose what I thought would show my best results. In fact, my supervisor has said I should continue seeing him regarding my results after the completion of my thesis, where he said he would be willing to also work on it and we can produce a solid publication out of it.

Things didn't always go smoothly though. As someone with little programming experience, I had a lot of difficulty obtaining numerical results initially. But as the project progressed I got much better on producing my own code to solve my problems. In August, I suffered a few unexpected delays, with my supervisor going away for two weeks and me having to sit an exam. This caused the project to overrun into time where I would have started a full time job, which made it a lot trickier to meet the initial deadlines I had set out for myself. But in the end I feel I got there in the end (just about!).

Marking myself against assessment criteria

- Student initiative and amount of guidance required

  I feel I was able to steer the project in a suitable direction producing my own results often on my own accord, and checking up with my supervisor at appropriate times to make sure it was a sustainable direction

- Scientific quality: how well the material in the project scope has been covered, e.g. are methods being used correctly, have results been discussed critically

  I feel the methodology is explained a lot throughout this paper. It is very heavy on the equations, which I have repeated myself on multiple occasions to ensure they're done correctly. I have also been equally critical of the models produced and throughout the paper it is looking to build upon this

- Breadth: the amount of background material, over and above lecture module content, that the student has incorporated into the project

  Due to the aforementioned departure from the material taught on my course, I felt it was necessary to perform a full review on queuing theory and optimisation methods to ensure the reader will fully understand the terminology and methods being used throughout. This required souring from a lot of external material to ensure it gave enough information.

- Originality: the extent to which the student has contributed own ideas, perspectives etc to the project

  Despite given an initial model and method form my supervisor to work with, throughout the process i was always going back and formulating it to make it as realistic as possible within the time frame I had

- Presentation and logical structure of the report, including English style, readability and coherence of the report as a whole, quality of the abstract, introduction, figures and reference list and the way the latter two are co-ordinated with the text.

  The project has a clear structure to it with each chapter building upon the foundations set by the previous ones in order to reach the final goal

Overall Mark: I am expecting to receive a strong Distinction in this project, but it comes down to how well I have managed to express my results and whether the reader gains as much as I did