Optimization of Call Centre Scheduling using the Cross-Entropy Method

Hjalmar Heimbürger  
\textit{hhei@kth.se}

Pontus Resare  
\textit{presare@kth.se}

SA104X Degree Project in Engineering Physics, First Level  
Department of Mathematics, Optimization and Systems Theory Division  
Royal Institute of Technology (KTH)  
Supervisor: Per Enqvist

May 28, 2014
Abstract

This thesis project investigates how the cross-entropy method can be applied to the optimization of scheduling problems. It is an important question because lower staffing leads to lower operating cost. Scheduling problems cannot be solved exactly in reasonable time, and finding methods to solve them efficiently is of great importance.

The cross-entropy method has been used with good results for other problems with time-consuming properties. The idea behind the cross-entropy method is to generate a random sample with possible solutions, and calculate how well they performed. Then that information is used to generate a new random sample with greater probabilities around good solutions. The method provides a formula how to update the probability distributions from which the random samples are generated.

The method can be applied to create a schedule for a call centre. However, it is uncertain how good the solutions are, what complexity the method has, and how differences in the problem specification will affect the method’s performance.

Sammanfattning


The Cross-Entropy method har tidigare använts för andra problem som också är väldigt tidskrävande, varför den möjligtvis skulle kunna användas på schemaläggning. Idén bakom metoden är att, utifrån en given sannolikhetsfördelning, generera möjliga lösningar och beräkna hur bra de är. Därefter genereras nya lösningar, fast denna gång med högre sannolikhet där de bra lösningarna var. Metoden ger också formler för hur dessa sannolikhetsfördelningar skall uppdateras.

Metoden kan användas för schemaläggning av ett call centre. Dock är det osäkert hur bra lösningarna är, vilken komplexitet problemet har och hur skillnader i problemspecifikationen påverkar resultatet.
# Contents

1. **Introduction**  
   1.1 Outline of the report ............................................. 2

2. **Optimization of Call Centre Scheduling**  
   2.1 Properties of a Call Centre .................................... 3
   2.2 Queueing Theory ..................................................... 4
   2.3 Problem formulation ............................................. 6

3. **The Cross-Entropy Method**  
   3.1 Background ............................................................. 9
   3.2 Cross-entropy applied to optimization .......................... 11
   3.3 Minimization of Cross Entropy .................................. 12
   3.4 Summary of Method ............................................. 13
   3.5 An example .......................................................... 14

4. **Results and Discussion**  
   4.1 Results ................................................................. 16
   4.2 Improvements on the model ....................................... 18
   4.3 Possible Improvements of the Method .......................... 18
   4.4 Conclusion .......................................................... 19

A. **Properties of the discrete Laplace distribution** ............... 20
Chapter 1

Introduction

The call centre industry is rapidly growing [1], and it is important to choose the staff-scheduling optimally, to minimize the cost while still maintaining good service and short waiting times. Any business that can do this will have a big advantage compared to those who cannot. To decide what shifts that the staff should work, and assign a suitable number of staff to each of them is therefore of great interest.

The cross-entropy method was developed around 1997 by Reuven Y. Rubinstein [2]. The idea was first proposed as an adaptive algorithm to find rare-event probabilities, based on the idea of minimizing the cross-entropy between two probability distributions. It turned out that the method had a much broader area of application. For instance, it could easily be adapted to tackle combinatorial optimization problems, such as the well known Travelling Salesman problem [2], and The Vehicle Routing problem with Stochastic Demands [3], which are problems that are time-consuming to solve, just like this problem.

The method focuses on generating random samples from a given distribution, and then updating the aforementioned distribution based on how well the different samples performed. The aim is to assign higher probability where the samples performed better, and smaller probability where the performance was worse. The key is to generate enough samples to make it converge to a good solution, but not too many as the computational time increases with the number of random samples.

The aim of this report is to try and apply this cross-entropy method to a scheduling problem, in particular scheduling of a call centre. Choosing another scheduling problem could alter the method in a major way, since there are vast differences between different types of scheduling problems [4].

1.1 Outline of the report

The report starts with [Chapter 2] which includes some background about scheduling problems in general, and call centre scheduling in particular. Some queueing theory will be included, which is needed to model the call centre. This is mainly for the unfamiliar reader. The scheduling problem at hand is also defined. [Chapter 3] concentrates on the cross-entropy method, the theory behind it, and how to apply it to the scheduling problem at hand. In [Chapter 4] the results of the simulations will be presented. These results will be the basis for the discussion in the second part of the chapter. Lastly, the conclusions of the paper will be presented.
Chapter 2

Optimization of Call Centre Scheduling

Scheduling problems emerge in a wide variety of settings, such as airline companies, when they need to assign staff to their flights, call centres, when they need to assign agents to answer calls, and fire-fighters, when they need to have a certain number firemen on call at all times. Several businesses depend greatly on optimizing these problems such that the cost is minimized. Personnel wages is a big part of the operating cost. In a call centre, the personnel cost is about 60-70% of the operating cost \cite{5}, and if it can be minimized, it saves a lot of money. A businesses that requires this usually is quite large, which means that the amount saved if the personnel cost is decreased with just a percent is enormous \cite{1}. For instance, if major airlines save 1% in crew costs, the savings amount up to tens of millions of dollars per year \cite{6}.

The biggest problem in solving scheduling problems is that the problems are NP-hard \cite{1}, and very time consuming to solve. Finding the optimal solution(s) is often impossible. Therefore, many methods are based on heuristics and the objective is usually to find solutions that are “good enough”.

2.1 Properties of a Call Centre

A call centre is an office that receives incoming calls from customers wanting some kind of support. There are several different classifications for call centres. In particular, the distinction between a single-skill and a multi-skill call centre is vital. In a single-skill call centre, all agents have the same skill-set, whereas in a multi-skill call centre, the agents can have different skill-sets.

Example: A call centre does customer support for a computer and phone company. It handles both hardware and software support. The agents have different skill sets. Type 1 agents are able to do hardware and software support for phones only, whereas type 2 agents are able to do hardware support for both phones and computers, but they cannot help with software support. This is a multi-skill call centre. These are harder to model, because it requires skill-based routing.

Skill-based routing is to decide who answers the incoming call. Depending on how many agents of a certain type that are busy, the incoming call gets routed to different agents. For instance, in the example above, if a customer calls with a hardware problem on their phone, and the type 1 agents are a lot busier than the type 2 agents, then the call would get routed to a type 2 agent.

In modelling a call centre, the random nature of the problem has to be taken into account. Calls arrive according to some, usually very complicated, stochastic process. There are several other random properties of the call centre. For instance, the duration of the calls and how long a customer stays in the queue before abandoning are random. To describe the queue in a call centre, queueing theory has to be used.
2.2 Queueing Theory

The general idea behind queueing theory can be summarized in the following way:

- Customers arrive with some intensity $\lambda$. This means that, in average, it arrives $\lambda$ customers per units of time.
- The customer can either receive service instantaneously or have to wait in a queue.
- If the customer has to enter the queue, it can either wait until it receives service, or leave the system without getting service.
- Customers are serviced with an average rate of $b = 1/\mu$ units of time per customer.
- The customers leave the system.

An illustration of this can be seen in Figure 2.1.

A call centre is a good example of a queueing system. Customers call to the call centre with intensity $\lambda$. They are serviced by $c$ agents, which each have a service rate of $1/\mu$. After they have been serviced, they hang up and leave the system. If no server is available, a queue forms.

2.2.1 Kendall’s notation

A queueing system is usually described with a more concise notation, known as Kendall’s notation.

$$A/B/c,$$

where

- $A$ is the arrival process;
- $B$ is the service time distribution;
- $c$ is the number of service stations.

The arrival process

The arrival process is a point process, which is a stochastic description of how the customers arrive to the system. There are several arrival processes that are common. These are:

- $M$ denotes a Poisson point process. In a Poisson point process, the time at which the customers arrives are independent, and exponentially distributed with identical rate parameter $\lambda$. $M$ comes from “Markov process” or “memoryless”. This is because the Poisson point process is a Markovian process.
- $D$ denotes a deterministic point process. There is no randomness in the arrival of the customers.
- $GI$ denotes a “general independent” point process, which means the time between arrivals are independent and identically distributed, and follow some general distribution.
- $G$ denotes a general point process.

The arrival process is usually assumed to be a Poisson point process. In reality, this might be a bad approximation. For instance, a simple store might be modelled as a queueing system, with the arrival process being a Poisson point process. However, if there are many customers in the store, fewer new customers will probably enter. In a call centre, however, before calling, the customers does not know the size of the queue, and this is not a complication.
Service time distribution

The service time is also stochastic. Each customer takes different amount of time to serve. The service times are assumed to be independent and identically distributed. Common distributions are:

- M denotes that the service times are exponentially distributed, they are $\text{Exp}(\mu)$ that is. Again, the $M$ stands for Markov, since the exponential distribution is memoryless, just like Markov processes.
- D denotes deterministic. There is no randomness involved.
- G denotes general, which means all service times follow some general distribution.

Number of service stations

The number of service stations is how many agents that can attend to customers at the same time. Figure 2.2 shows this for $c = 4$. In optimizing call centre scheduling, larger $c$ implies higher costs. Therefore, $c$ want to be kept to a minimum, while still maintaining a proper level of service.

Other parameters

Sometimes, there are other parameters included in Kendall’s notation. These include the size of the queue, how big the sample population is that can enter the queue, and the queues discipline, for example first in, first out or last in, first out. For a call centre, the only reasonable assumptions is that the size of the queue is infinite, the sample population is infinite as well, and the discipline of the queue is first in, first out.

2.2.2 Queueing systems

M/M/$c$-system

A M/M/$c$-system is usually the first approximation to make when modelling a call centre. This means that the arrival process is a Poisson point process with intensity $\lambda$, the service times are exponentially distributed with intensity $\mu$, and there are $c$ servers. There are certain properties of an M/M/$c$ queue that are interesting. What is the probability that a customer has to wait longer than a certain time? Does the customer need to queue, and if it has to, how long will the expected time be? These properties are: [7]

\[
P(\text{Customer needs to queue}) = C_c = \frac{\frac{(cp)^c}{c!}}{(1 - \rho)\left(\sum_{i=0}^{c-1} \frac{(cp)^i}{i!} + \frac{(cp)^c}{c!(1 - \rho)}\right)}
\]

\[
\text{E[Time in queue]} = \frac{C_c \rho}{\lambda(1 - \rho)}
\]

\[
P(\text{Time in queue} \leq \tau) = 1 - C_c e^{-\rho(1 - \rho)\tau}, \quad \tau > 0
\]
Here, $\rho = \frac{\lambda}{c\mu}$ is the server utilization, which is what fraction of the time the servers are busy. For the queue to not grow infinitely, $\rho < 1$. If $\rho > 1$, then customers will arrive faster than the servers can handle, and the queue will grow without limit.

### 2.2.3 M/G/c-system

Changing the service time distribution from exponentially distributed to a general distribution is often a better model. This would probably be necessary to investigate if this method is to be applied in real life, since the exponential model have shortcomings. However, this is not within the scope of this paper, and will not be investigated.

### 2.2.4 Simplifications in the model

A lot of simplifications of the model has been done in this paper, as the focus has been on the cross-entropy method to solve the problem. Firstly, the queue is a M/M/c queue. Secondly, a customer has unlimited patience. This means that they will never hang up the phone. This assumption is not favourable when applying to real life situations, as it tends to lead to over-staffing and the model will yield longer queues compared to what is observed, even if the abandonment rate is low [8].

### 2.3 Problem formulation

The aim is to choose staffing $x$ for a call centre, such that the cost $S(x)$ is minimized, while still maintaining a pre-defined service level to assure that customers get service fast, and that the queue is short.

During one day, people call to the call centre. The day is split into periods, where each period is a time interval. For instance, one period could be the 30 minute interval between 9:00 and 9:30. For any given period $i$, the intensity of incoming calls is $\lambda_i$ [calls per units of time]. The agents have a service time of $1/\mu$ [units of time per call].

There is a predefined number of possible shifts. These decide when to start and end the day, as well as when the breaks are. The aim is to decide the number of staff working each shift to minimize the cost. All the possible shifts are described by the matrix $Q$. It is a binary matrix with elements $q_{ij}$. If $q_{ij} = 1$, shift $j$ works during period $i$, and $q_{ij} = 0$ otherwise.

**Example:** There are 2 shifts, both working period 1, and shift 1 working period 2 but not 3, and shift 2 working period 3 but not 2, then

$$Q = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ q_{31} & q_{32} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

There are $x = (x_1, x_2, \ldots, x_m)^T$ members of staff working during one day. $x_j$ is the number of staff members working shift $j$ during one day. It is obvious that $x$ needs to be non-negative integers, since neither $-1$ or 3.5 number of persons can work a shift. The number of staff working during period $i$, denoted $y_i$, is

$$y = Qx$$

The cost for each period is given by $c = (c_1, c_2, \ldots, c_p)^T$, which, for a certain staffing $x$, gives the cost $S(x)$

$$S(x) = c^TQx$$

Worth noting is the assumption that the arrival intensities are constant over these periods, and that the queue reaches its steady state much faster than the length of the period. This might not always be correct. However, if it does not reach its steady state, the formulas from section 2.2 will not be valid, and thus we cannot use them for the constraints.
For any given period, there are certain service levels that need to be maintained. Let \( g(x) \) be the service level. It is usually defined as

\[
g(x) = \frac{W_\tau}{W_\infty + A_\tau}
\]

where \( t \) is the time spent waiting, \( W_\tau = \mathbb{E}[\# \text{ of customers waiting } t \leq \tau] \), and \( A_\tau = \mathbb{E}[\# \text{ of customers abandoning after } t > \tau] \). If the customers have infinite patience, and therefore never abandon, then \( A_\tau = 0 \). Let \( \xi \) denote the service level that needs to be maintained. By using (2.7), the constraint can be formulated as

\[
g(x) = \frac{W_\tau}{W_\infty + A_\tau} = \mathbb{P}(\text{Time in queue } t \leq \tau) \geq \xi
\]

For a given service level, it is possible to calculate how many persons that are needed to assure that it is maintained.

**Example:** A call centre receives incoming calls following a Poisson process with intensity \( \lambda = 50 \) calls per minute. The call lengths are exponentially distributed with mean service time \( 1/\mu = 2 \) minutes per call. The call centre has a goal of answering 80% of the calls within 1 minute. (2.3) gives that with \( c = 103 \), 85% of the calls get answered within 1 minute, and thus satisfying the service level criterion. Therefore, (2.8) can either be left as it is, or can be recalculated as a constraint for the number of staff during each period by using (2.3). There is advantages with both choices. The latter is less accurate, since some information about how much the service level constraint is broken is lost. However, the computation time is much faster with the latter. In this report, the service level constraints are recalculated to staffing demand for each period. Let \( d \) denote the number of staff needed each period. Using (2.3), together with (2.5), to recalculate (2.8) as the number of staff needed each period, the constraint can be formulated as

\[
Qx \geq d
\]

Then, (2.6) and (2.9) gives the following optimization problem

\[
\min_x \quad S(x) = c^T Qx \quad \text{s.t.} \quad Qx \geq d \quad x \geq 0 \quad x \in \mathbb{Z}
\]

**2.3.1 Soft Constraints**

The Cross-Entropy method, which will be used to solve (2.10), is a stochastic method. It is based on adjusting a probability distribution to generate random, feasible solutions. The first constraint in (2.10) adds several complications. This is due to the fact that the constraint is being enforced on \( Qx \) rather than \( x \) (\( x \) is what is generated). Adjusting a probability distribution to only generate feasible solutions while still enforcing this constraint is impossible. This difficulty will be dealt with by making it a soft constraint.

A soft constraint is when a constraint is introduced as a cost in the objective function, penalizing if the constraint is broken. The size of the penalty is usually decided by to what extent the constraint is broken. Of course, the penalty can be \( \infty \), and thus making any feasible solution to the original problem a better solution. However, this is undesirable in this case, since the optimal solution will be close to the boundary of the feasible set. A solution that is close to the optimal solution might be infeasible. Nevertheless, if it is close enough, it should still be used to update the probability distributions. Therefore, a soft constraint that penalizes how much the constraint is broken is essential.

Let \( h(Q, x, d) = (h_1, \ldots, h_p) \) be defined as how much the constraint is broken by

\[
h_i(Q, x, d) = \begin{cases} 
(d - Qx)_i & \text{if } (Qx)_i < d_i, \\
0 & \text{otherwise}
\end{cases}
\]
Let $a_i$ be the penalty in period $i$, and using this along with $h(Q, x, d)$ (2.10), can be reformulated as

$$
\begin{align*}
\min_x \quad & S(x) = c^T Q x + a^T h(x, Q, d) \\
\text{s.t.} \quad & x \geq 0 \\
& x \in \mathbb{Z}
\end{align*}
$$

(2.12)
Chapter 3

The Cross-Entropy Method

Sections 3.1 and 3.2 in this chapter is based on “The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning” by Rubinstein and Kroese [2].

Consider the following minimization problem:

\[
\min \ S(x) \\
\text{s.t.} \quad x \in \mathcal{X}
\]  

(3.1)

If \( \mathcal{X} \) is finite, a naive way to solve this is to test all \( x \in \mathcal{X} \). This would be problematic if the size of \( \mathcal{X} \) is big. A better way to find the optimal solution is to generate a few feasible solutions from a given probability distribution. Then depending on how good the solutions are, change the probability distribution such that the good solutions gets a higher probability, and the bad solutions gets a lower probability compared to before. This is the general idea of the Cross-Entropy method, (in this text it will be abbreviated as the CE-method). This method was invented in order to estimate probabilities of rare events, where it has been very effective. It will first be shown how the CE-method can be applied to rare event estimation, and afterwards how it could be applied to solve the call centre problem.

3.1 Background

In problems where finding an analytical expression of a probability is hard, one way to estimate the probability is to generate a big random sample and use the outcomes as an estimation. For instance, consider the problem of calculating the probability of some function \( S(X) \) being smaller than \( \gamma \), where the random variable \( X \) has the probability density function \( f(x; u) \). This can be written as:

\[
\ell = \Pr_u(S(X) \leq \gamma) = \mathbb{E}_u[I_{\{S(X) \leq \gamma\}}]
\]  

(3.2)

Here \( I \) is the indicator function which is 1 if \( S(X) \leq \gamma \) and 0 otherwise. If (3.2) cannot be found explicitly, it can be estimated by generating a sufficiently large number of random \((X_1, \ldots, X_N)\) samples from \( f(x; u) \).

\[
\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(X_i) \leq \gamma\}}
\]  

(3.3)

This is called Monte-Carlo simulation. However, if \( S(X) \leq \gamma \) is a rare event, this method is not very effective. In order to get a accurate estimation, a very large number of samples have to be generated. A way to solve this problem is to use importance sampling. The idea is to draw the random samples from a different probability density \( g(x) \) where \( S(X) \leq \gamma \) is more common, and compensate this with a
weight function.

\[
\ell = \mathbb{E}_u \left[ I(S(x) \leq \gamma) \right] = \int I(S(x) \leq \gamma) f(x; u) dx = \int I(S(x) \leq \gamma) \frac{f(x; u)}{g(x)} g(x) dx =
\]

\[
= \mathbb{E}_g \left[ I(S(X) \leq \gamma) \frac{f(X; u)}{g(X)} \right] \tag{3.4}
\]

This gives the new estimator of (3.2)

\[
\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I(S(X_i) \leq \gamma) \frac{f(X_i; u)}{g(X_i)} \tag{3.5}
\]

Here, the random samples \(X_1, \ldots, X_N\) are generated from the new probability density function \(g(x)\).

How effective this method is depends on how \(g(x)\) is chosen. Therefore, an algorithm to generate \(g(x)\) is needed. Let \(g^*(x)\) denote the optimal probability density function. If it is optimal, all the \(X_1, \ldots, X_N\) that are generated from \(g^*(x)\) fulfills the criterion \(S(x) \leq \gamma\). \(g^*(x)\) is given by:

\[
g^*(x) = I(S(x) \leq \gamma) f(x; u) \ell \tag{3.6}
\]

By substituting (3.6) into (3.5)

\[
\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I(S(X_i) \leq \gamma) \frac{f(X_i; u)}{I(S(X_i) \leq \gamma) f(X_i; u) \ell} = \frac{1}{N} \sum_{i=1}^{N} \ell = \frac{N\ell}{N} = \ell \tag{3.7}
\]

which proves that \(g^*(x)\) is the optimal probability density function. However, there is a problem with using the \(g^*(x)\) defined in (3.6). In order to get the best probability density \(g^*(x)\), \(\ell\) needs to be known, and that was what we were searching for!

In order to find \(g^*(x)\), an iterative algorithm will be used. The key to this algorithm is the realization that the random samples \(X_i\) that fulfill \(S(x) \leq \gamma\) can be seen as outcomes of a probability density \(g(x)\) which should be quite close to \(g^*(x)\). If a new probability \(h(x)\) is created that is similar to \(g(x)\), the random samples from this density should fulfill \(S(x) \leq \gamma\) more often. However, in order to obtain the new probability \(h(x)\), an expression for the similarity between probability densities has to be obtained.

### 3.1.1 Kullback-Liebler divergence

There are several ways to define how similar two densities are, in this case \(g(x)\) and \(h(x)\). One definition which has been shown to be convenient, is the Kullback-Leibler divergence, also known as cross-entropy, which in this text is going to be abbreviated as CE. The Kullback-Leibler divergence is defined as:

\[
\mathcal{D}(g, h) = \mathbb{E}_g \left[ \ln \frac{g(X)}{h(X)} \right] = \int g(x) \ln g(x) dx - \int g(x) \ln h(x) dx \tag{3.8}
\]

The CE can be seen as a “pseudo-distance” because it fulfills some of the properties of a distance; it always non-negative \(\mathcal{D}(g, h) \geq 0\) and \(\mathcal{D}(g, h) = 0\) if and only if \(g(x) = h(x)\). However, it does not fulfill the property of being symmetric (\(\mathcal{D}(g, h) \neq \mathcal{D}(h, g)\)).

The aim here is to generate \(h(x)\) as similar to \(g(x)\) as possible. This is the same as minimizing the CE between the probability density functions in (3.8).

\[
\min \mathcal{D}(g, h) = \min \left(\int g(x) \ln g(x) dx - \int g(x) \ln h(x) dx \right) \tag{3.9}
\]

Let \(h(x)\) belong to the same family of distributions as \(f(x; u)\) and denote it as \(f(x; v)\). The probability density function \(g(x)\) is constant and the parameter \(v\) is the variable. Therefore minimizing the cross entropy is the same as maximizing the second term in (3.9):

\[
\min \mathcal{D}(g, f) = \max_v \int g(x) \ln f(x; v) dx \tag{3.10}
\]
Substituting $g(x)$ in (3.10) with the optimal density in (3.6) gives

$$\min \mathcal{D}(g, f) = \max_v \frac{\int I_{\{S(x) \leq \gamma\}} f(x; u)}{\ell} \ln f(x; v) dx$$  \hspace{1cm} (3.11)

Let $v^*$ be the parameter that minimizes the CE between $f(x; v)$ and $g^*(x)$, then

$$v^* = \arg\max_v \frac{\int I_{\{S(x) \leq \gamma\}} f(x; u)}{\ell} \ln f(x; v) dx$$  \hspace{1cm} (3.12)

Because $\ell$ is constant it is not needed in the expression and (3.11) simplifies to

$$v^* = \arg\max_v \int I_{\{S(x) \leq \gamma\}} f(x; u) \ln f(x; v) dx = \arg\max_v \mathbb{E}_u \left[ I_{\{S(X) \leq \gamma\}} \ln f(X; v) \right]$$

By using the technique of importance sampling again and introducing a new probability density function $f(x; w)$, this becomes

$$v^* = \arg\max_v \int I_{\{S(x) \leq \gamma\}} \frac{f(x; u)}{f(x; w)} \ln f(x; v) f(x; w) dx$$  \hspace{1cm} (3.13)

$$= \arg\max_v \mathbb{E}_w \left[ I_{\{S(X) \leq \gamma\}} \frac{f(X; u)}{f(X; w)} \ln f(X; v) \right]$$  \hspace{1cm} (3.14)

This can be estimated by

$$v^* = \arg\max_v \frac{1}{N} \sum_{i=1}^N I_{\{S(X_i) \leq \gamma\}} \frac{f(X_i; u)}{f(X_i; w)} \ln f(X_i; v)$$  \hspace{1cm} (3.15)

where $X_1, \ldots, X_N$ are generated from $f(x; w)$. By using $f(x; w)$ as the previous probability density function this can be turned into an iterative method. One of the things that makes the Cross-Entropy method practical is that it is often possible to obtain the $v$ analytically. If the probability densities are sufficiently nice, the updating formulas can be calculated by differentiating with respect to $v$ and setting it equal to zero.

$$\frac{1}{N} \sum_{i=1}^N I_{\{S(X_i) \leq \gamma\}} \frac{f(X_i; u)}{f(X_i; w)} \nabla \ln f(X_i; v) = 0$$  \hspace{1cm} (3.16)

### 3.2 Cross-entropy applied to optimization

In the previous section the main CE-method was described. In this section it will be shown how to apply the CE-method to discrete optimization. Let $S(x)$ be a function that should be minimized under the constraint that $x \in \mathcal{X}$. This can be written as:

$$S(x^*) = \gamma^* = \min_{x \in \mathcal{X}} S(x)$$  \hspace{1cm} (3.17)

The aim is to find $x^*$. In order to be able to use the CE-method, an associated stochastic problem is defined in the following way:

$$\ell(\gamma) = \mathbb{P}_u(S(x) \leq \gamma) = \mathbb{E}_u[I_{\{S(X) \leq \gamma\}}]$$  \hspace{1cm} (3.18)

In the CE-method for rare events, new probability density functions $f(x; v)$ are chosen in succession in order to make $\ell$ more probable. If $\gamma$ is chosen close to $\gamma^*$ solving of associated problem with the rare event method should make the new probability density functions to start to converge towards $x^*$.

The updating rules for a discrete optimization can be obtained by using (3.16) from the previous section, this time with:

$$\frac{f(X_i; u)}{f(X_i; w)} = 1$$  \hspace{1cm} (3.19)
This is because in this case $\ell$ is not interesting and (3.19) is just a weight term, which is used to normalize the expression. Inserting (3.19) into (3.16), and introducing the iteration counter $t$, the following updating rules are obtained

$$
\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(S(X_i) \leq \gamma_t) \nabla \ln f(X_i; v_t) = 0 \quad (3.20)
$$

where the $X_1, \ldots, X_N$ are obtained from the probability density function $f(x; v_{t-1})$ of the previous iteration.

The main difference between rare event estimation and optimization, is that in the latter case the algorithm depends on both $\gamma_t$ and $v_t$. Choosing suitable $\gamma_t$:s is very important. The usual way to solve this is to define a percentile $\rho$, sort all the cost functions $S(1) \leq S(2) \leq \ldots \leq S(N)$ and set $\gamma_t = S(N\rho)$. With this definition each $\gamma_t$ should be smaller or equal to the one from the previous iteration, and $f(x; v_t)$ should converge towards $x^*$.

### 3.3 Minimization of Cross Entropy

#### 3.3.1 Selection of distribution

Now it has to be decided what probability distribution which the random solutions should be generated from. Selecting this probability distribution is not trivial. There are several important properties of the distribution that needs to be fulfilled. First of all, it needs to have a location parameter and a scale parameter. The location parameter is needed to center the distribution around any given value, and the scale parameter is needed to make it center all mass at that value. Secondly, an analytical expression to update the parameters via (3.20) needs to be obtained. Therefore, a distribution with a “nice” expression for the probability mass function is needed. There are also several properties decided by the constraints in the problem. Since the number of people on each shift needs to be non-negative integers, a discrete distribution with semi-infinite support ($x \geq 0$) is needed.

A probability distribution that meets all these criteria is a discrete version of the Laplace distribution that is truncated at $k = 0$ to ensure the semi-infinite support. It has the following probability mass function:

$$
f(k; \mu, p) = \frac{1 - p}{p\mu - \mu + p^{|k-\mu|}} p^{|k-\mu|} \quad (3.21)
$$

See Appendix A for more detailed information about the distribution.

#### 3.3.2 Approximations

To minimize the Cross-Entropy, (3.20) has to be solved. However, this is troublesome, because it is impossible to find an analytical expression for both $\mu$ and $p$. Solving it numerically would be possible, but it would make the method much slower. This is not a viable option, since the problem is already time-consuming as it is. Therefore, a few approximations have been made in order to obtain an analytical solution. Firstly, if $\mu$ is assumed to be integer, the denominator in (3.21) simplifies to

$$
f(k; \mu, p) = \frac{1 - p}{1 + p - p^{\mu+1}} p^{|k-\mu|} \quad (3.22)
$$

This approximation is still not enough, as it is still impossible to obtain a general formula. However, if $p^{\mu+1} \ll 1$, it could be simplified to

$$
f(k; \mu, p) = \frac{1 - p}{1 + p} p^{|k-\mu|} \quad (3.23)
$$

This is the same expression as for the discrete Laplace distribution with infinite support. This is expected, since $p^{\mu+1} \ll 1$ either means that $\mu \gg 1$, or $p \ll 1$ (or both), which in turn means that almost all probability mass (in the infinite case) is for non-negative values.
3.3.3 The updating formulas

Now that the probability distribution has been selected, the expressions that minimize the cross-entropy can be calculated. For a given set of data \( \mathbf{X}_i = (k_{i1}, k_{i2}, \ldots, k_{im}) \), the probability of it happening is given by the product of the probability of each individual event happening

\[
f(\mathbf{X}_i; \mathbf{\mu}, \mathbf{p}) = \prod_j f(k_{ij}; \mu_j, p_j) = \prod_j \frac{1 - p_j}{1 + p_j} |k_{ij} - \mu_j|
\]

(3.24)

\[
\ln f(\mathbf{X}_i; \mathbf{\mu}, \mathbf{p}) = \sum_j \ln f(k_{ij}; \mu_j, p_j) = \sum_j \ln \left( \frac{1 - p_j}{1 + p_j} |k_{ij} - \mu_j| \right)
\]

(3.25)

The updating formulas for \( \mu_j \) can be obtained by inserting (3.25) into (3.20), where the derivative is with respect to \( \mu_j \)

\[
\frac{\partial}{\partial \mu_j} \sum_i I_{\{S(\mathbf{X}_i) \leq \gamma\}} \ln f(\mathbf{X}_i; \mathbf{\mu}, \mathbf{p}) = \frac{\partial}{\partial \mu_j} \sum_i \sum_j I_{\{S(\mathbf{X}_i) \leq \gamma\}} \ln \left( \frac{1 - p_j}{1 + p_j} |k_{ij} - \mu_j| \right)
\]

(3.26)

\[
= \frac{\partial}{\partial \mu_j} \sum_i \{ I_{\{S(\mathbf{X}_i) \leq \gamma\}} [\ln(1 - p_j) - \ln(1 + p_j) + |k_{ij} - \mu_j| \ln p_j] \}
\]

\[
= \sum_i \left[ I_{\{S(\mathbf{X}_i) \leq \gamma\}} \left( |k_{ij} - \mu_j| \frac{1}{p_j} - \frac{1}{1 - p_j} - \frac{1}{1 + p_j} \right) \right]
\]

\[
= \frac{1}{p_j(1 - p_j)} \sum_i \left[ I_{\{S(\mathbf{X}_i) \leq \gamma\}} \left( |k_{ij} - \mu_j| (1 - p_j^2) - 2p_j \right) \right] = 0
\]

(3.27)

Similarly for \( p_j \), (3.25) is inserted into (3.20), but this time, the derivative is with respect to \( p_j \)

\[
\frac{\partial}{\partial p_j} \sum_i I_{\{S(\mathbf{X}_i) \leq \gamma\}} \ln f(\mathbf{X}_i; \mathbf{\mu}, \mathbf{p}) = \frac{\partial}{\partial p_j} \sum_i \sum_j I_{\{S(\mathbf{X}_i) \leq \gamma\}} \ln \left( \frac{1 - p_j}{1 + p_j} |k_{ij} - \mu_j| \right)
\]

(3.28)

\[
= \frac{\partial}{\partial p_j} \sum_i \{ I_{\{S(\mathbf{X}_i) \leq \gamma\}} [\ln(1 - p_j) - \ln(1 + p_j) + |k_{ij} - \mu_j| \ln p_j] \}
\]

\[
= \sum_i \left[ I_{\{S(\mathbf{X}_i) \leq \gamma\}} \left( |k_{ij} - \mu_j| \frac{1}{p_j} - \frac{1}{1 - p_j} - \frac{1}{1 + p_j} \right) \right]
\]

\[
= \frac{1}{p_j(1 - p_j)} \sum_i \left[ I_{\{S(\mathbf{X}_i) \leq \gamma\}} \left( |k_{ij} - \mu_j| (1 - p_j^2) - 2p_j \right) \right] = 0
\]

(3.29)

Let \( L := \sum_i I_{\{S(\mathbf{X}_i) \leq \gamma\}} \) and \( K_j := \sum_i I_{\{S(\mathbf{X}_i) \leq \gamma\}} |k_{ij} - \mu_j| \). Then the solution to (3.29), which is a simple 2nd degree polynomial, can be written as

\[
p_j = \begin{cases} 
0 & \text{if } K_j = 0 \\
\frac{L}{K_j} + \sqrt{1 + \left( \frac{L}{K_j} \right)^2} & \text{otherwise}
\end{cases}
\]

(3.30)

### 3.4 Summary of Method

**Main Algorithm**

1. Set \( t = 1 \) and choose the start parameters \( \mathbf{\mu}_0 \) and \( \mathbf{p}_0 \)
2. Generate \( N \) random samples of \( \mathbf{X} \) according to probability distribution \( f(\mathbf{X}; \mathbf{\mu}_{t-1}, \mathbf{p}_{t-1}) \)
3. Calculate the performance function \( S(\mathbf{x}) \) by using (3.31)
4. Sort the performance of each sample \( S_{(1)} \leq \ldots \leq S_{(N)} \) and let \( \gamma_t = S_{(N \rho)} \)
5. Use updating rule from equation (3.32) and equation (3.33) to calculate new parameters \( \mu_t \) and \( p_t \) respectively.

6. If the stop criterion is met, go to step 7, otherwise set \( t = t + 1 \) and return to step 2.

7. Let \( x^* = \arg\min_{x} S(x) \)

8. Return \( x^* \)

The cost \( S(x) \) for the staffing \( x \) (more detailed explanation in (2.12)).

\[
S(x) = c^T Q x + a^T h(x, Q, d) \tag{3.31}
\]

The algorithm for updating \( \mu_j \)

\[
\mu_j = \text{median}\{k_{ij} : S(X_i) \leq \gamma\} \quad j = 1, \ldots, m \tag{3.32}
\]

The algorithm for updating \( p_j \)

\[
p_j = \begin{cases} 
0 & \text{if } K_j = 0 \\
- \frac{L}{K_j} + \sqrt{1 + \left( \frac{L}{K_j} \right)^2} & \text{otherwise}
\end{cases} \quad j = 1, \ldots, m \tag{3.33}
\]

where \( L := \sum_i I_{\{S(X_i) \leq \gamma\}} \) and \( K_j := \sum_i I_{\{S(X_i) \leq \gamma\}} |k_{ij} - \mu_j| \).

### 3.5 An example

The idea of the method is perhaps easiest to grasp by studying an example of the algorithm. Let \( S(x) \) be the blue graph seen in Figure 3.1 and the objective is to minimize \( \min_{x \in \mathbb{R}} S(x) \) (3.34)

The Laplace distribution was used (see Appendix A.1 for probability density function) to generate random solutions. These are the dots in the graph. The black dots are the good solutions that are used for updating the parameters in the distribution.

In the first graph, the data is quite spread out. Comparing plot one to plot two, it is easy to see that where the solutions are generated has changed. The density of the generated solutions around the minimum is much higher in the second graph. For the third and fourth iteration, almost all probability mass is at the minimum, which it converges to after two more iterations (it takes two more iterations mainly because the stop criterion was quite strict).

\[\text{The function grows as } x^4 \text{ for big } |x| \text{ and there is no minima outside the graph.}\]
Figure 3.1: Trying to find the $x$ that minimizes the blue plot. $N = 1000$, $\rho = 0.01$
Chapter 4

Results and Discussion

4.1 Results

In all plots, the cost has been normalized to $c = 1$. The penalty constant for understaffing (unless stated otherwise) is $a = 15$. The method has solved all problems it has been given. In Figure 4.1a, the demand is specified to test the method. There is only one shift, shift 7, that works during periods 1, 9, 11 and 17. This means that if the method works, it should put all staff on that shift. As seen in Figure 4.1a, it is needed 100 people on shift 7, and the method finds the optimal solution.

In Figure 4.1b, the method does not converge towards the optimal solution. The solution which it converges to is good, but it could be better. For instance, switching one person from shift 9 to shift 20 would eliminate the overstaffing during periods 1, 9, 16 and 17, and reduce the understaffing during period 11 to only 1 person. This is probably due to the existence of several local minima, and the method has a hard time finding the global minimum. This is in contrast to Figure 4.1a, where the only minimum that exists is a global minimum.

Figure 4.2 shows the importance of choosing the penalty constant for understaffing properly. If the penalty is too small, it will provide a solution with major understaffing during some periods. The penalty has to be sufficiently large, such that the cost of understaffing during one period is larger than what it costs to overstaff during the other periods. Let the length of the shortest shift be 12 periods. If the penalty is chosen to be $a = 3$, the cost of understaffing one period is 3, whereas the cost of overstaffing all other periods is 3. The method therefore interprets the understaffing as the better

![Figure 4.1a: Demand constructed such that there exists only one shift that covers all the periods in the demand (shift 7).](image1)

![Figure 4.1b: The number of persons needed are constant throughout the day.](image2)

Figure 4.1: Two different simulations.
solution. In Figure 4.2a, there is no understaffing, whereas the understaffing in Figure 4.2b is quite severe.

There is, however, two sides of the coin. If $a$ was chosen to be very large, it did tend to provide solutions with more overstaffing. Although this overstaffing was usually quite small. The extra overstaffing is probably due to there being a steep change in the cost on the infeasible side of the boundary. Then the solution it converges to will be shifted away from the boundary.

The method has performed well overall. However, it has not been compared to other methods of solving the same problem. Therefore, how well it has performed is hard to judge solely based on this data.

### 4.1.1 Choosing parameters properly

The complexity and the effectiveness of the method was surveyed. The result can be seen in Table 4.1 and Table 4.2. The service demand is identical to the one in Figure 4.2. All the data is averaged over 10 iterations. As can be seen in the tables, choosing parameters properly is essential. For instance, in Table 4.2 when $N = 10^2$ and $\rho = 0.01$, the method does not converge at all. This is because $\rho N = 1$, meaning that only 1 sample is used to update the parameters. Secondly, when $N = 10^3$ and $\rho = 0.1$, the solutions were bad, which implies that $\rho N$ should not be too big. This occurred in both Table 4.1 and 4.2. The problem seems to be of complexity $O(N)$, and the number of iterations until the stop criterion was met seems to be independent of both $N$ and $\rho$. Consequently, the time until the stop criterion was met also seems to be $O(N)$. When the number of possible shift and the number of periods increased with factors 6.5 and 3.2 respectively, the computation time only increased with a factor 4. Due to this fact, the conclusion is that the computation time of the problem grows slower than $O(size(Q))$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\rho$</th>
<th>Minimum when stop criterion is met</th>
<th>Iterations until stop criterion is met</th>
<th>Time until stop criterion is met [s]</th>
<th>Minimum after 100 iterations</th>
<th>Time after 100 iterations [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>0.1</td>
<td>2001</td>
<td>26</td>
<td>0.162</td>
<td>2037</td>
<td>0.445</td>
</tr>
<tr>
<td>$10^2$</td>
<td>0.01</td>
<td>2075</td>
<td>35</td>
<td>0.164</td>
<td>2066</td>
<td>0.369</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.1</td>
<td>2038</td>
<td>41</td>
<td>0.579</td>
<td>2025</td>
<td>1.290</td>
</tr>
<tr>
<td>$10^3$</td>
<td>0.01</td>
<td>2028</td>
<td>42</td>
<td>0.581</td>
<td>2021</td>
<td>1.204</td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.1</td>
<td>2040</td>
<td>36</td>
<td>3.629</td>
<td>2034</td>
<td>9.736</td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.01</td>
<td>2021</td>
<td>42</td>
<td>4.070</td>
<td>2020</td>
<td>9.157</td>
</tr>
</tbody>
</table>

Table 4.1: Data from simulations where the size of $Q$ is $17 \times 27$. Stop criterion is when the minimum has not changed for 10 iterations.
4.2 Improvements on the model

There are several assumptions made in the modelling stage of this report that are questionable if it was to be implemented in a real life situation. The queue being an M/M/c queue is an okay first estimation, and the assumption that the arrival process is a Poisson point process is good. However, the arrival intensity can probably not be considered to be constant during 15 or 30 minute intervals [9]. Therefore, the assumption that the queue reaches its steady state for the period fast is problematic. If the arrival intensity is changing over time, the queue does not have a steady state distribution, and all the queueing theory formulas (2.1)-(2.3) will not be valid, and more thorough analysis how the queue will behave is needed. However, this assumption is much better than the assumption that the service times are exponentially distributed. Research suggests that the service times follow the log-normal distribution [9]. It is very different from the exponential distribution, and assuming log-normal distribution would probably alter the staffing demands quite dramatically.

Another, not very realistic, assumption is that customers have infinite patience. As Jouini et al. [8] notes, not taking abandonments into account causes pessimistic queueing times and overstaffing. There are several improvements possible. One possible improvement is to add a probability of a customer hanging up if they do not receive service immediately. Another would be introduce customer patience according to some probability distribution, for instance exponential [10].

The possible shifts should be chosen in a way that there is a big variety in shifts available. This means that there should be no periods where many of the shifts work, as it will most likely lead to overstaffing during those periods. The ideal distribution on shifts is probably when around 50 % of the shifts work during any given period.

4.3 Possible Improvements of the Method

There are some problems with the algorithm that could be improved. In many cases the solutions start to oscillate between two or more good solutions. There are three possible explanations to the oscillating behaviour; problems with the updating formula for $p$, problems with the updating formula for $\mu$, or problems with the MATLAB implementation.

In Chapter 3, some approximations was done in order to find updating formulas for $\mu$ and $p$. These approximations are only valid when $\mu_j \gg 0$ or $p_j \ll 1$. This is not the always case.

It does not seem to be possible to find an explicit updating expression for $\mu_j$ when the conditions $\mu_j \gg 0$ and $p_j \ll 1$ are not fulfilled. However, using the approximation that $\mu_j$ being an integer led to the following expression:

$$f(k; \mu, p) = \frac{1 - p}{1 + p - p^{\mu+1}}p^{k-\mu}$$  \hspace{1cm} (4.1)$$

By using (4.1) in (3.20), a polynomial of degree $\mu + 2$ is obtained. This means that if $\mu = 0, 1, 2$, this equation can be solved analytically for $p$. Solving these cases separately would give modified updating formulas for $p$, which would give more accurate updating formulas.
Another possible solution for solving the oscillating behaviour is to use a discrete Laplace distribution with infinite support instead, and using a soft constraint to impose the non-negativity constraint on \( x \). If this is done, the updating formulas in (3.32) and (3.33) will be correct, instead of just being approximations.

Another problem is that the understaffing penalty \( a \) has to be chosen heuristically. There is a possibility that the penalty parameter \( a \) could be obtained by calculations instead of heuristic methods. A good first guess of \( a \) could probably be calculated from the demand vector \( d \) and the shift matrix \( Q \). It is worth noting that choosing \( a \) to small is a larger problem than choosing \( a \) to big.

### 4.4 Conclusion

Firstly, the problem of scheduling a call centre, given that the service level constraints are recalculated as staff constraints, can be solved with the Cross-Entropy method. Secondly, if all constraints on the schedule can be converted to costs, the Cross-Entropy method can probably be used to optimize the schedule of a more realistic call centre. However, it is uncertain how different staff constraints and problem sizes affects how good the solutions will be. It is also uncertain how the complexity of the Cross-Entropy method compares to other methods.
Appendix A

Properties of the discrete Laplace distribution

The aim of this chapter is to define a discrete random variable that can be seen as a discrete version of the Laplace distribution.

Laplace distribution

The Laplace distribution has the following probability density function

\[ f(x; \mu, \sigma) = \frac{1}{2\sigma} \exp \left( -\frac{|x - \mu|}{\sigma} \right) \]  

(A.1)

The discrete Laplace distribution

It is possible to define a discrete version of the Laplace distribution. For any discrete random variable, the probability mass function can be written as

\[ P(X = k) = \frac{f(k)}{\sum_{k' \in \mathcal{K}} f(k')} \]  

(A.2)

Inserting (A.1) into (A.2), along with the substitution \( p = e^{-1/\sigma} \), yields

\[ P(X = k) = \frac{p^{k-\mu}}{\sum_{k' \in \mathcal{K}} p^{k'-\mu}} \]  

(A.3)

Infinite support

If \( \mathcal{K} = \mathbb{Z} \), then \( X \) has the following probability mass function and cumulative distribution function

\[ f(k; \mu, p) = P(X = k) = \frac{1 - p^{k-\mu}}{1 + p} \]

\[ F(k; \mu, p) = P(X \leq k) = \begin{cases} \frac{1}{1 + p^{k-\mu}} & \text{if } k \leq \mu \\ \frac{1}{1 + p^{k-\mu}} & \text{if } k > \mu \end{cases} \]  

(A.4)
Semi-infinite support

If $K = \mathbb{N}$, the probability mass function and cumulative distribution function will change the normalization constant.

$$P(X = k) = \frac{1}{Z} p^{k-\mu} \quad Z = \sum_{k=0}^{\infty} p^{k-\mu}$$ \hspace{1cm} (A.5)

The sum can be found analytically by splitting it into two geometric series.

$$Z = \sum_{k=0}^{\infty} p^{k-\mu} = \sum_{k=0}^{|\mu|-1} p^{\mu-k} + \sum_{k=|\mu|}^{\infty} p^{k}$$

$$= p^{\mu} \sum_{k=0}^{|\mu|-1} (p^{-1})^{k} + p^{-|\mu|} \sum_{k=|\mu|}^{\infty} p^{k}$$

$$= p^{\mu} \frac{1 - p^{-|\mu|}}{1 - p^{-1}} + p^{-|\mu|} \frac{1}{1 - p}$$

This yields the probability mass function

$$f(k; \mu, p) = p^{k-\mu} \left/ \left( \frac{p^{|\mu|}-\mu + p^{1-(|\mu|)-\mu} - p^{\mu+1}}{1 - p} \right) \right.$$ \hspace{1cm} (A.7)

If $\mu$ is an integer, (A.7) simplifies to

$$f(k; \mu, p) = p^{k-\mu} \left/ \left( \frac{1 + p - p^{\mu+1}}{1 - p} \right) \right.$$ \hspace{1cm} (A.8)

If $\mu \gg 1$, then $p^{\mu+1} \to 0$, and it can be even more simplified to

$$f(k; \mu, p) = \frac{1 - p}{1 + p} p^{k-\mu}$$ \hspace{1cm} (A.9)

This is the same equation as (A.4).

Cumulative Distribution Function

Finding the cumulative distribution function for the semi-infinite discrete Laplace distribution is very similar to the methodology to finding the probability mass function.

$$F(k; \mu, p) = \mathbb{P}(X \leq k) = \sum_{k'=0}^{k} \frac{1}{Z} p^{k'-\mu}$$ \hspace{1cm} (A.10)

Since the exponent in (A.10) is dependent on $|k' - \mu|$, it is easiest to divide it into two cases

$$k \leq \mu : \quad F(k; \mu, p) = \frac{1}{Z} \sum_{k'=0}^{k} p^{\mu-k'} = \frac{p^{\mu}}{Z} \sum_{k'=0}^{k} p^{-k'} = \frac{p^{\mu}}{Z} \frac{1 - p^{-(k+1)}}{1 - p}$$

$$= \frac{p^{\mu} \left( p^{-k} - p \right)}{Z (1 - p)}$$ \hspace{1cm} (A.11)

$$k > \mu : \quad F(k; \mu, p) = 1 - \mathbb{P}(X > k) = 1 - \frac{1}{Z} \sum_{k'=k+1}^{\infty} p^{k'-\mu}$$

$$= 1 - \frac{p^{k-\mu+1}}{Z} \sum_{k'=0}^{\infty} p^{k'} = 1 - \frac{p^{k-\mu+1}}{Z} \frac{1}{1 - p}$$
Since the cumulative distribution function $F(x) = U$ is non-decreasing, it is possible to find an inverse $x = F^{-1}(U)$, which then can be used in the Inverse-Transform Method \[2\] to generate random variables following this distribution.

$$
\begin{align*}
\text{k} \leq \mu : & \quad U = F(k; \mu, p) = \frac{p^\mu p^{-k} - p}{Z} \frac{1}{1 - p} \\
\quad & \Leftrightarrow p^{-k} = Z(1 - p)U^{p^{-\mu}} + p \\
\quad & \Leftrightarrow k = -\frac{\ln[Z(1 - p)U^{p^{-\mu}} + p]}{\ln p}
\end{align*}

\begin{align*}
\text{k} > \mu : & \quad U = F(k; \mu, p) = 1 - \frac{p^{k-\mu+1}}{Z} \frac{1}{1 - p} \\
\quad & \Leftrightarrow p^k = Z(1 - U)(1 - p)p^{\mu-1} \\
\quad & \Leftrightarrow k = \frac{\ln[Z(1 - U)(1 - p)p^{\mu-1}]}{\ln p}
\end{align*}

Results

$$
\begin{align*}
f(k; \mu, p) &= \frac{1 - p}{p^{\lceil \mu \rceil - \mu + p^{1 - (\lceil \mu \rceil - \mu)} - p^{\mu+1}}} p^{k-\mu} \\
F(k; \mu, p) &= \begin{cases} 
\frac{p^\mu p^{-k} - p}{Z} \frac{1}{1 - p} & \text{if } k \leq \mu \\
1 - \frac{p^{k-\mu+1}}{Z} \frac{1}{1 - p} & \text{if } k > \mu
\end{cases}
\end{align*}

k = \begin{cases} 
-\frac{\ln[Z(1 - p)U^{p^{-\mu}} + p]}{\ln p} & \text{if } k \leq \mu \\
\frac{\ln[Z(1 - U)(1 - p)p^{\mu-1}]}{\ln p} & \text{if } k > \mu
\end{cases}$$
References


