Caching Over-the-Top services in an ISP's network

Stefan Almind Jensen
Networks and Distributed Systems
Master Thesis - 14gr1020

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:
Caching Over-the-Top services in an ISP's network

Theme:
Network planning

Project Period:
Spring Semester 2014

Project Group:
14gr1020

Participant(s):
Stefan Almind Jensen

Supervisor(s):
Jose Manuel Gutierrez Lopez
Michael Jensen

Copies: 4

Page Numbers: 101

Date of Completion:
June 2, 2014

Abstract:

Due to over-the-top (OTT) content utilising approximately 67% of the fixed access downstream Internet traffic during peak periods in North America, OTT content providers have deployed caches internally in various ISP networks. This thesis presents a method for planning caches for OTT services in an ISP's network to provide an offload to the backhaul link. This offload is found by minimising the maximum link load. Good deployments of caches are found using a Genetic Algorithm (GA) with a routing algorithm and genetic operators that are highly targeted at solving the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem. As input, Geographical Information System (GIS) data from Bornholm, Denmark, is used to generate realistic results. The demand and caches specifications are provided through a case study of Netflix. Results are given for the before and after picture of deploying caches, a cost-fitness trade-off analysis and a numerical evaluation of the link loads for both cache and gateway traffic. The results show that the proposed method provides a highly offloaded (link load balanced) network.

Preface

This report is the documentation of a master thesis project, on the 4^{th} semester of the Master's programme; Networks and Distributed Systems, Aalborg University. The thesis focuses on planning an amount and location of caches to offload an Internet service provider (ISP)'s backhaul, thereby providing better avoidance of congestion.

Using a GA to solve the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem, good cache deployments resulting in an offload of the backhaul are found within a reasonable amount of time. The report documents an introduction, motivating why there exists a problem with congested links, and why caches are used to solve this problem. A problem formulation defines the problem using knowledge of facility location (FL) problems and defines the used two topologies constructed from Bornholm. An analysis contains a discussion of how the LLB-CFL can be solved and why a GA is used to solve it. The analysis also contains a case study of Netflix, where information on demand and cache are used. Finally the analysis contains a thorough review of the assumptions and delimitations performed. A chapter for the proposed method contains a thorough explanation of the input data, why specific genetic operators are used and efficient choices to their parameters. A description of a cost-fitness trade-off analysis details why it may be preferable to analyse which amount of caches that yield the best trade-off between cost of solution and offload of links. A numerical evaluation of the case when demand is serviced through both caches and gateways allows a network planner to identify which links can potentially become congested and may require additional capacity. Lastly results are presented, showing a clear picture of the method providing offloaded ISP networks for different amounts of caches.

A DVD is attached, containing *.pdf*-files for online literature (*/literature*) and the repository of the report (*/report*) and code (*/code*).

Aalborg University, June 2, 2014

Stefan Almind Jensen
<saje09@student.aau.dk>

Contents

Chapter 1

Introduction

This chapter introduces the motivation for the study of caching OTT content in an Internet service provider's network and presents a problem statement, summarising what problem this master thesis seeks to solve. A section containing expected contributions thoroughly describes the aspects of this project and relates them to the study curriculum.

1.1 Motivation

In recent years the popularity of over-the-top (OTT) content and its accessibility from a wide range of devices has resulted in a large utilisation of Internet traffic. OTT content incorporates video, audio and other types of media, where the concept of OTT refers to content that is not provided by network operators. A report from Sandvine [2013] shows that OTT real-time entertainment (RTE) (streaming video and audio) is the most dominant traffic category for fixed access in North America. RTE accounts for approximately 67% of the fixed access downstream traffic during peak periods, as is seen in figure 1.1.

An investigation of what individual applications contribute to the majority of the aggregated Internet traffic during peak periods is seen in figure 1.2. This shows that Netflix and YouTube are the two dominating peak period applications, both being video streaming services. These specific OTT services have achieved immense popularity over recent years with a collective delivery of billions of hours of streamed content each month [Empson, 2013][Youtube, 2014].

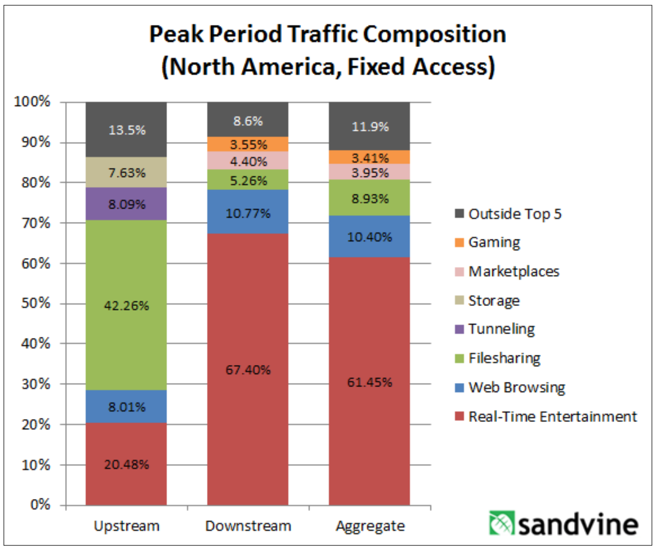
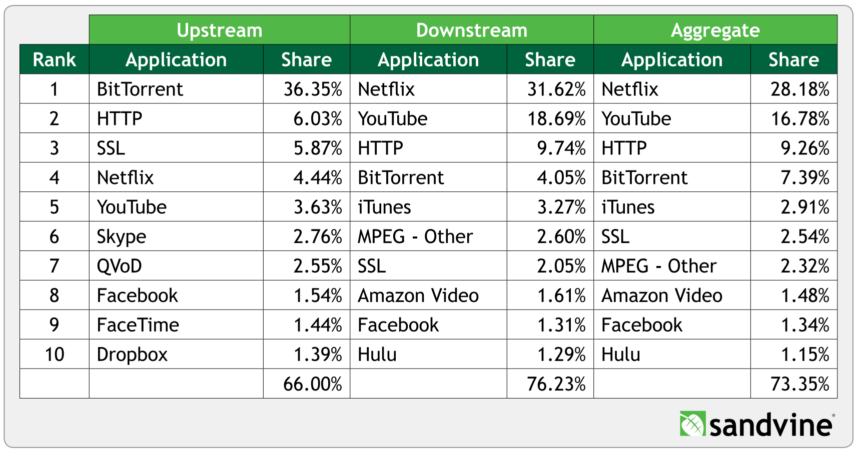


Figure 1.1: Peak Period Aggregate Traffic Composition - North America, Fixed Access [Sandvine, 2014, p. 5]



	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	36.35%	Netflix	31.62%	Netflix	28.18%
2	HTTP	6.03%	YouTube	18.69%	YouTube	16.78%
3	SSL	5.87%	HTTP	9.74%	HTTP	9.26%
4	Netflix	4.44%	BitTorrent	4.05%	BitTorrent	7.39%
5	YouTube	3.63%	iTunes	3.27%	iTunes	2.91%
6	Skype	2.76%	MPEG - Other	2.60%	SSL	2.54%
7	QVoD	2.55%	SSL	2.05%	MPEG - Other	2.32%
8	Facebook	1.54%	Amazon Video	1.61%	Amazon Video	1.48%
9	FaceTime	1.44%	Facebook	1.31%	Facebook	1.34%
10	Dropbox	1.39%	Hulu	1.29%	Hulu	1.15%
		66.00%		76.23%		73.35%

sandvine

Figure 1.2: Top 10 Peak Period Applications - North America, Fixed Access [Sandvine, 2013, p. 6]

Now that OTT content providers, such as Netflix and Google (YouTube), have become increasingly popular, certain networking problems arise for the content providers, since large amounts of data must traverse the Internet. In a design of OTT content delivery networks (CDNs) it is of great importance that a quality of service (QoS) is guaranteed, thereby providing a good service performance and user experience. This problem is not only solved through a technical design of a CDN capable of handling the capacity and other QoS requirements, but also through peering agreements as to secure service availability for all subscribers.

Netflix and Google, who runs YouTube, have both built their own CDNs to achieve a larger control of their content delivery and to provide content at minimal cost [Rayburn, 2014]. These CDNs include deployment of caches in an ISP's network to reduce congestion. The main task of the caches is to supply the most popular content to an ISP's customers. Less popular content (tail content) and fill traffic is then provided by servers in the CDN via peering. The planning of caches for OTT services is what this project seeks to contribute to. A problem statement will in the following describe what problem this project focuses on.

1.2 Problem statement

As teased in the motivation, this project will focus on the problem of cache planning for OTT services in an ISP's network. Caches ensure a reduction in network distance to OTT content to achieve a higher QoS and can provide an offload of an ISP's backhaul[1]. Planning of these caches mainly consists of deciding the amount and location of caches, which heavily influences how OTT traffic propagates through an ISP's network. Knowledge of the cache's characteristics is essential for the planning of caches and consists of e.g. capacity constraints and content availability. A cache plan for this project consists of a specification for deployment of a sufficient amount of caches to supply the demand from all households within a specific area.

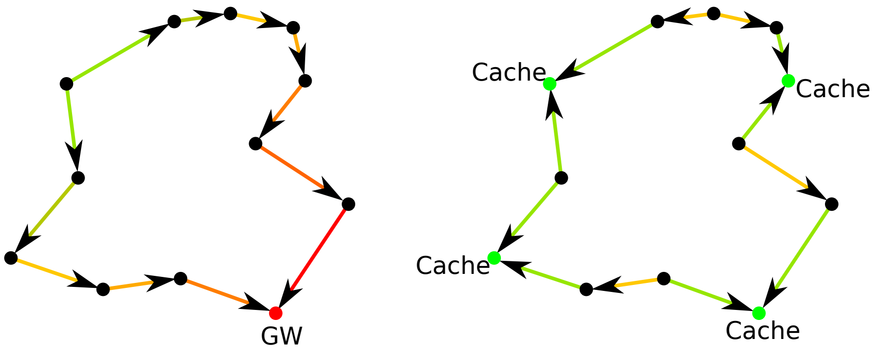


Figure 1.3: Differences in link loads when content is accessed through gateways and from caches. Link loads are coloured from green to red, indicating respectively the least to the most loaded link.

By estimating traffic flows, through a choice of cache locations, an evaluation of how well a cache deployment plan is at providing an offload of the backhaul network can be performed. The optimal cache deployment plan is the one which results in the most balanced network load. By balancing the load of OTT traffic across an ISP's network, the amount of available bandwidth on backhaul links is increased and the probability of congestion occurring is decreased. Figure 1.3 shows the possibility of decreasing link loads, when content is accessed from caches instead of gateways. The optimal solution would of course be to deploy as many

[1]A set of links in the ISP's core/backbone network

caches as possible, but a cost of deploying a cache must be considered, as there is a trade-off between the cost of caches and when the backhaul is sufficiently offloaded. By considering these described aspects of the project in the planning of caches for OTT content, a problem statement is formulated:

> “When ISP networks display a large amount of OTT network traffic and exhibit congested links, how can caches be planned in an ISP's network to offload the backhaul network and provide a better QoS for OTT content subscribers?”

1.3 Expected contributions

This section explains the expected contributions given the problem statement. The expected contributions are presented in comparison with the study curriculum of the master thesis of Networks and Distributed Systems. This includes obtaining knowledge, skills and competencies.

Knowledge

According to the study curriculum, the student:

K1 has knowledge, at the highest international level of research, of at least one of the core fields of the education

K2 has comprehension of implications of research (research ethics)

K1 is fulfilled through the investigation of state of the art methods for network planning and the planning of caches (facility location). Through this investigation, methods relating to the same conditions as this problem can be used to solve the problem. Other methods not yet used in the specific field of the problem can also be used to solve the problem. The problem and its conditions can in more simple terms be described as finding a method for balancing the network load in networks by deploying caches with capacity constraints.

This project covers a subject that is currently meeting a lot of media attention regarding net neutrality and where costs are placed for cache deployments in ISP networks. Net neutrality is being discussed, because many think that OTT content is experiencing degraded QoS due to throttling by the ISP, which is not necessarily the case. It is also discussed in relation to the question if the OTT content providers are also given prioritised traffic flows when deploying caches. Costs are also being debated in regards to the question if OTT content providers should pay additional costs (other costs than those related to cache hardware/software) for the placement of caches in the ISP's network, e.g. rent. This project does not affect these subjects, as the resulting plan for cache deployment does not require prioritised traffic flows to achieve an offload of an ISP's network and does not propose a business structure defining which parties are billed for cache deployments. This knowledge does on the other hand show that the project has a comprehension of the circumstances that the caching solutions are deployed under and therefore fulfilling **K2**.

Skills

According to the study curriculum, the student:

- **S1** is able to reflect on a scientific basis on their knowledge,
- **S2** can argue for the relevance of the chosen problem to the education including specifically account for the core of the problem and the technical connections in which it appears
- **S3** can account for possible methods to solve the problem statements of the project, describe and assess the applicability of the chosen method including account for the chosen delimitation and the way these will influence on the results of the product
- **S4** can analyse and describe the chosen problem applying relevant theories, methods and experimental data
- **S5** is able to describe the relevant theories and methods in a way that highlights the characteristics and hereby document knowledge of the applied theories, methods, possibilities and delimitations within the relevant problem area
- **S6** has the ability to analyse and assess experimental data, including the effect the assessment method has on the validity of the results.

The knowledge attained in this project, as described earlier, is reflected on scientifically, by relating it to previous knowledge within networking and acquiring/investigating empirical evidence such as link loads (**S1**). The problem of cache planning directly relates to the study of Networks and Distributed Systems (**S2**) as it includes subjects as:

- Traffic theory and modelling
- GIS data
- Graph theory
- Wired networks
- Network management

These subjects will be covered when e.g. network load is modelled, routes are generated, caches are placed at viable locations within a graph of the ISP's network and subscribers of an ISP are associated to caches. Point **S3**, **S4**, **S5** and **S6** are fulfilled by looking at the problem statement. The problem of this project is an optimisation problem of finding the best amount and location of caches. Exact solutions like e.g. branch-and-bound (linear programming) may be a solution for the problem, but due to the running time of such algorithms, a meta-heuristic algorithm is expected to be able to solve the problem within a reasonable amount of time and providing good local optimal solutions. Metaheuristic algorithms also have a low complexity allowing for a more simple design in solving the problem. A metaheuristic algorithm finds a viable solution within a defined solution space that seeks to maximise/minimise an objective function according to some constraints while evaluating candidate solutions until a stopping criterion. GIS data associated with Bornholm, Denmark, is used to construct a scenario with network termination (NT) populations and a given network architecture, which is used as input for the optimisation algorithm. The optimisation is performed on an objective function that will indicate how well a given network is balanced. A candidate solution is a cache deployment plan conforming to the constraints of the caches' capacities. A metaheuristic algorithm will result in a solution that is deemed optimal, as there is no guarantee for global optimality when using metaheuristic algorithms. The optimality of a solution will be deemed by how well the backhaul network is offloaded and the cost of deploying caches. Designing a method for finding such a deployment of caches is expected to be the main contribution of this project. After a satisfactory deployment is found it is numerically evaluated using knowledge of cache characteristics, including aspects such as content availability (how much traffic is serviced through gateways). While the planning focuses on a network where all requests from all households are serviced from caches inside the ISP's network, a numerical evaluation focuses on getting a better picture of actual traffic flows, when content may not be available in the caches and traffic must leave the network through gateways. This evaluation will result in knowledge of how load is distributed in the network when not all requests are serviced by the caches, but instead through gateways.

Competencies

According to the study curriculum, the student:

C1 is able to communicate scientific problems in writing and orally to specialist and non-specialist.

C2 is able to control situations that are complex, unpredictable and which require new solutions.

C3 is able to independently initiate and to perform collaboration within the discipline and interdisciplinary as well, and to take professional responsibility,

C4 is able to independently take responsibility for his or her own professional development and specialization.

This report will document an analysis, method and present results, which will be defended at an oral examination (**C1**). It is shown through the following report that an understanding of various subjects is achieved and a link between them has been formed to construct a method for cache planning (**C2**). Through discussions with the industry and supervisors the report shows a focus and understanding of the background for this thesis(**C3**). The student's main expected contribution is to present a viable method for cache planning for OTT services and support this method with empirical data and results (**C4**).

Chapter 2

Problem formulation

This chapter includes a description of what field the problem is associated to and a generic definition of the problem, which is supported with examples. The problem formulation also includes a presentation of a real-world scenario, which is used to find a good deployment of caches according to some realistic constraints and an objective function.

2.1 The LLB-CFL problem

The Link Load Balanced Capacitated Facility Location (LLB-CFL) problem is presented in the following. As there is no current research into the link load balanced variation of the FL problem, a definition of the problem and a method for solving it will be the contribution of this thesis. The following sections present the FL problem, the capacitated variation and lastly the link load balanced variation.

2.1.1 Facility location

The problem of this thesis lies within the field of combinatorial optimisation, specifically facility location (FL). FL problems are not easily solvable, as they are NP-hard problems [Li and Yeh, 2005]. The quote below clearly describes what a FL problem seeks to solve. The terms presented in this quote are describe in the following section.

> “*Facility location problems locate a set of facilities (resources) to minimize the cost of satisfying some set of demands (of the customers) with respect to some set of constraints.*”
>
> *Farahani and Hekmatfar [2009]*

Facility
A facility maps to the definition of a cache. A number of caches that must be deployed at a subset of the given CO locations, also called potential facility locations. There must be sufficient caches deployed to supply the demand from each CO.

Demand
The demand in the network originates at each CO location. The demand at a CO is assumed to be equal to the NT population associated to it. These demands are gathered from a geographical area in Denmark, which are presented later in section 2.3. This means that this thesis will apply the problem to GIS data to give a real-world picture of how the demands of a population can be satisfied with a cache deployment that provides a good offload of an ISP's backhaul.

Cost
The cost of satisfying the demand is given by an objective function. This objective function uses GIS data in the form of population sizes and a given CO interconnection to evaluate how load balanced the links are for a specific cache deployment.

Constraints
Constraints define the wanted or feasible solutions (cache locations) in a solution space. In the case of cache planning this includes constraints to the amount of caches wanted, cost of a solution and the maximum capacity of a cache.

2.1.2 Capacitated variation

In this thesis the amount of sustainable throughput to a cache is limited. This variation of the FL problem is called the capacitated facility location (CFL) problem, and describes a constraint to the amount of demand a facility can supply. This means that if a cache is full capacity, there must be additional cache to supply the remaining demand.

2.1.3 Link load balanced variation

The link load balanced variation is new and is formulated in this thesis. The idea of finding a deployment of caches that is link load balanced comes from the need to increase the available bandwidth of an ISP's backhaul to avoid congestion due to OTT traffic. Balancing the links generally means to find a distribution of traffic that increases the amount of available bandwidth. There are different ways to achieve link load balancing and increasing the available bandwidth, but they have different applications [Santos et al., 2009]. Load balancing is performed through minimisation of the maximum link load, which guarantees the amount of available bandwidth on all links in the network. This is the chosen objective, as this can provide robustness against unpredictable traffic growths and is better at avoiding congestion. This will in turn allow other services to occupy the link. Other methods of load balancing links is by finding a deployment of caches with the smallest average/summed link load. This only provides a guarantee to the amount of available bandwidth for the overall network.

This variation of the problem only increases complexity to the already hard problem of FL, since it adds requirements to the type of solution provided by any algorithm that solves it. These types of solutions are link load balanced solutions, and the algorithm that solves the link load balanced variation, must be able to distinguish between solutions by how well this objective is solved. It will additionally require more than just simple route calculations for assigning demand to facilities, but more realistic estimations of routing within ISP networks, specifically choices to how and when these demands are assigned to the facilities.

One can examine the maximum link load utilisation, U, with $0 \leq U \leq 1$. This will give a general idea of what is achieved by minimising the maximum link load. When the objective is to minimise the maximum link load utilisation, then all demand in the network can increase with a factor, i_f, corresponding to $(1-U)/U$ [Santos et al., 2009]. The increase corresponds to the relation as seen in equation 2.1. This is of course assuming the increase occurs uniformly and has the same origin-destination pairs. The increase factor, i_f, shows how robust the network is at handling unpredictable traffic growth or in other words how good it is at avoiding congestion. Figure 2.1 depicts this relation.

$$i_f = \frac{1-U}{U} \tag{2.1}$$

$$1 = U + U \cdot i_f \tag{2.2}$$

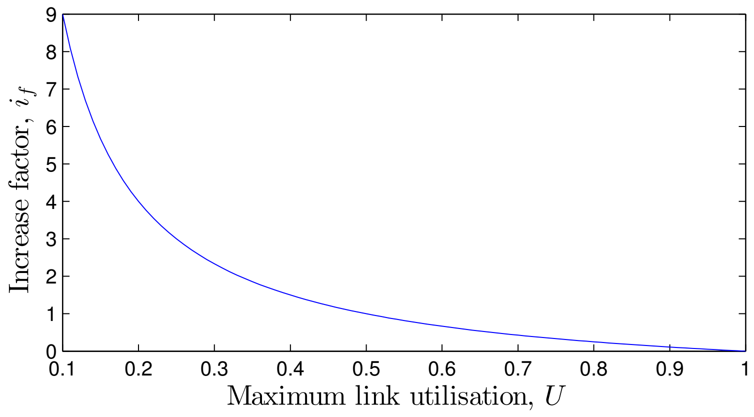


Figure 2.1: Allowed increase to the utilisation of a link

A generic definition of the optimisation problem is presented in the following section. This definition is supported using an example of an ISP network and different potential solutions in the form of cache deployments. The definition includes mathematical notation as it gives a clear picture of the problem space for the inputs and outputs of the designed algorithm.

2.1.4 Generic definition

This section introduces notation for the LLB-CFL problem. An ISP's network is described using an undirected graph $G_I = (V, X, Y, A, N)$ and is the input for an optimisation algorithm. V are a set of vertices corresponding to COs. A set of coordinate vectors (X, Y) describe the geographical locations of each CO. An adjacency matrix, A, represents the ISP's connectivity between COs. At each CO there are an number of associated households or NTs, described with a population vector, N. By passing an input graph to an optimisation algorithm, the algorithm must be able to find a cache deployment plan that best solves the problem according to the objective function.

A solution graph is described using a directed solution graph $G_S = (V, X, Y, A, N, S, P, C, L)$. Two potential solution graphs are presented in figure 2.2. The solution graph, G_S, includes all information on the input from G_I. The additional information of G_S consists among others of a vector of binary values, S, representing a cache being placed at a specific CO. This means that it is assumed that caches can only be placed at the same locations as COs. This is a logical assumption, since it will nullify the costs of new housing to place the caches within. The solution graph also consists of a list of paths, P, from each CO to a cache. The generation of a path is necessary information as it will allow for a calculation of the load of each link and cache. The cost associated with travelling along each of these paths to a cache is stored in the cost matrix, C. The last piece of information provided in the solution graph is L, a link and cache load matrix. This matrix represents the cache loads in the diagonal, while link loads are represented in all the other elements of the matrix. It is not a symmetric matrix, as traffic flows in a specific direction following a CO's path. For each solution there is a cost, *Cost(S)*, dependent on the number of caches deployed.

A maximum cache capacity (*CacheCap*), link capacity (*LinkCap*) and cost of solution (*RequiredCost*) define the constraints of the problem and are used to verify a solution's validity. These values respectively describe the maximum amount of users/bandwidth a cache can handle, the maximum amount of throughput any link can achieve and the exact cost a cache plan must have. Deploying more caches allows for less traffic to propagate across the network. If one rather wants to constrain the solution to a specific number of maximum deployed caches, instead of a cost of solution, a *CacheCost* of 1 can be attributed to each cache.

For some of the constraints, relaxation can be performed when they are exceeded, so they do not yield infeasible solutions, but instead result in a degradation of service quality. If the *LinkCap* or *CacheCap* constraints are exceeded, the quality of e.g. a video stream will be decreased to meet these constraints. If, on the other hand, the *RequiredCost* constraint is exceeded, it will become an infeasible solution, resulting in the solution being rejected.

A formal definition of the problem is now presented using mathematical notation. The definition introduces what is given (the input), what the output is and what constrains the solution space.

Given

- an undirected input graph $G_I = (V, A, P, C)$
- V : a set of COs, where
  $V \;=\; \{v_1, v_2, \ldots, v_n\}$ and
  $n \;=\; |V|$ (number of COs)
- (X, Y) : xy-coordinate vectors of CO locations, where
  $x_i, y_i \;\in\; \mathbb{N}^+$ and
  $i \;\in\; 1..n$
- A : an adjacency matrix with elements
  $a_{ij} \;=\; \begin{cases} 1, & \text{if connected} \\ 0, & \text{otherwise} \end{cases}$, where
  $i, j \;\in\; 1..n$
- N : a NT population (demand) vector with elements
  $n_i \;\in\; \mathbb{N}^+$, where
  $i \;\in\; 1..n$
- $CacheCost$: a cost of one cache
  $CacheCost \;\in\; \mathbb{N}^+$

The output is

- a directed solution graph $G_S = (V, X, Y, A, P, S, P, L)$
- S : an open caches vector with elements
  $s_i \;=\; \begin{cases} 1, & \text{if cache } i \text{ is open} \\ 0, & \text{if cache } i \text{ is closed} \end{cases}$, where
  $i \;\in\; 1..n$
- P : a matrix of list of paths, where each path is a sequence of edges alike
  $p_{14} \;=\; \{v_1, \ldots, v_4\}$, where
  $p_{ij} \;:\;$ a path originiting from v_i and ending in v_j and
  $i, j \;\in\; 1..n$
- C : a CO to cache association cost matrix with elements
  $c_{ij} \;=\; \begin{cases} \mathbb{R}^+, \text{if } s_j = 1 \\ 0, \text{otherwise} \end{cases}$, where
  $i, j \;\in\; 1..n$
- L : a link and cache load matrix with elements
  $l_{ij} \;=\; \begin{cases} \mathbb{N}^+, & \text{if link/cache is loaded} \\ 0, & \text{otherwise} \end{cases}$, where
  $i, j \;\in\; 1..n,$
  $l_{ij} \;=\; l_{ji},$
  $l_{ij} \forall i \neq j \;:\;$ link loads and
  $l_{ij} \forall i = j \;:\;$ cache loads
- $Cost(S)$: a cost of solution, where
  $Cost(S) \;=\; \sum_{i=1}^{n} (s_i) \cdot CacheCost$

Where the solution space is constrained by,

- $LinkCap$: a link capacity, where
  $l_{ij} \forall i \neq j \quad \leq \quad LinkCap$ and
  $LinkCap \quad \in \quad \mathbb{N}^+$
- $CacheCap$: a cache capacity, where
  $l_{ij} \forall i = j \quad \leq \quad CacheCap$ and
  $CacheCap \quad \in \quad \mathbb{N}^+$
- $RequiredCost$: a required cost of solution, where
  $Cost(S) \quad = \quad RequiredCost$

2.1.5 Objective function

The solution graph, G_S, is found by searching for a solution that provides a good offload and load balancing, but a value is required to describe how well this objective is achieved. A good load balance is found when a routing is found that minimizes the maximum link load, as was described earlier in section 2.1.3. The maximum link load is found using equation 2.3, where each element of the load matrix, l_{ij}, where $i \neq j$, represents a link load.

$$l_{max} = \max_{i \neq j}(l_{ij}) \tag{2.3}$$

For each potential deployment of caches a maximum link load is found. Choosing the best of these solutions is performed by finding the minimum of all maximum link loads. This is performed as is seen in equation 2.4.

$$min(\, l^1_{max} \, , \, l^2_{max} \, , \, \ldots \,) \tag{2.4}$$

2.2 Example

An example is formulated in the following to support the formal definition. The example contains an example ISP network and two potential solutions for a cache deployment. These are seen in figure 2.2. The example network is interconnected using a single ring topology with edge costs defined by distances. The link loads of the potential solutions are defined as the amount of NTs occupying the link. Likewise, the cache load is the amount of NTs it serves.

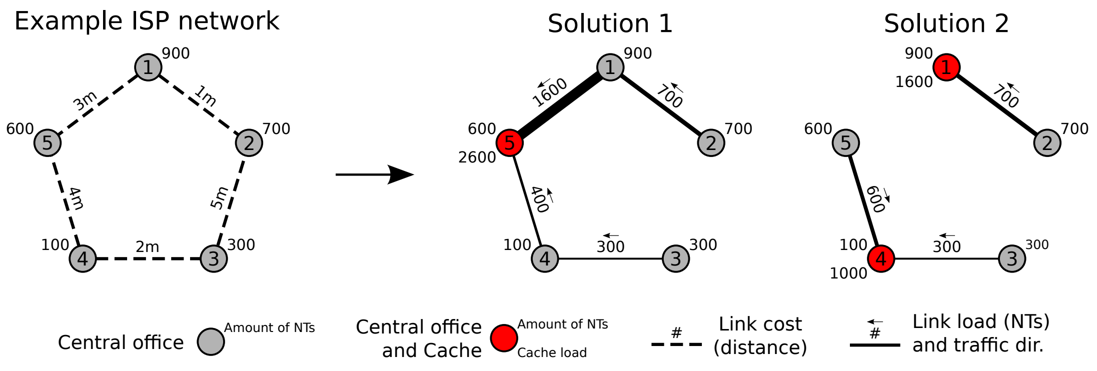


Figure 2.2: An example of a network is shown to the left, where interconnectivity and the amount of NTs associated to a CO are seen. The two solutions show choices to the amount and placement of caches resulting in different link loads, dependent on the paths the NTs follow from its associated CO to a cache. The solutions represent a link load by the number of NTs occupying a link and a cache load by the number of NTs it serves.

Initially the ISP's network is represented using the notation in the formal definition in example 2.1. Thereafter, the best solution is chosen given the two potential solutions and constraints to both the cache capacity and link capacity.

Example 2.1 (Example ISP network)
The example ISP network contains five COs and are represented as a set of vertices:

$$V = \{ v_1, v_2, v_3, v_4, v_5 \} \tag{2.5}$$

The network's adjacency matrix is written as

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.6}$$

which is the general form for single ring topologies. The population of NTs associated to each CO is represented with:

$$N = \begin{array}{c} \\ \end{array} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \\ \left[\; 900 \right. & 700 & 300 & 100 & \left. 600 \;\right] \end{array} \tag{2.7}$$

Lastly, the distances between COs are represented in the link cost matrix:

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 3 \\ 1 & 0 & 5 & 0 & 0 \\ 0 & 5 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 4 \\ 3 & 0 & 0 & 4 & 0 \end{bmatrix} \tag{2.8}$$

Example 2.2 and 2.3 show two potential solutions of the example ISP network. These are arbitrarily chosen and do not reflect solutions that are near any optimal solution.

Example 2.2 (Potential solution 1)
A cache deployment with one cache placed at vertex v_5 is represented with the vector:

$$\begin{array}{ccccccc} & & v_1 & v_2 & v_3 & v_4 & v_5 & \\ S^1 = & [& 0 & 0 & 0 & 0 & 1 &] \end{array}, \tag{2.9}$$

where a value of 1 represents a cache is placed at the location. Arbitrary paths are chosen (but should be generated using the cost and adjacency matrix) for each CO:

$$P_1^1 = \{ v_1, v_5 \},\ P_2^1 = \{ v_2, v_1, v_5 \},\ P_3^1 = \{ v_3, v_4, v_5 \},\ P_4^1 = \{ v_4, v_5 \},\ P_5^1 = \{ v_5 \} \tag{2.10}$$

These paths are used to calculate the link and cache loads seen both in the figure and in the following load matrix.

$$L^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1600 \\ 700 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 0 & 400 \\ 0 & 0 & 0 & 0 & 2600 \end{bmatrix} \tag{2.11}$$

Example 2.3 (Potential solution 2)
A cache deployment with two caches placed at vertices v_1 and v_4 are represented with the vector:

$$\begin{array}{ccccccc} & & v_1 & v_2 & v_3 & v_4 & v_5 & \\ S^2 = & [& 1 & 0 & 0 & 1 & 0 &] \end{array} \tag{2.12}$$

Arbitrary paths are chosen for each CO:

$$P_1^2 = \{ v_1 \},\ P_2^2 = \{ v_2, v_1 \},\ P_3^2 = \{ v_3, v_4 \},\ P_4^2 = \{ v_4 \},\ P_5^2 = \{ v_5, v_4 \} \tag{2.13}$$

These paths are used to calculate the link and cache loads seen both in figure 2.2 and in the following load matrix.

$$L^2 = \begin{bmatrix} 1600 & 0 & 0 & 0 & 0 \\ 700 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 600 & 0 \end{bmatrix} \tag{2.14}$$

Example 2.4 shows a validation of the potential solutions using the constraints to the load matrix and a choice of an optimal solution.

Example 2.4 (Validating the potential solutions and choosing the best)
With the following assumptions

$$LinkCap = 2000 \text{ NTs} \tag{2.15}$$
$$CacheCap = 2000 \text{ NTs} \tag{2.16}$$
$$CAPEX = 200 \text{ USD} \tag{2.17}$$
$$OPEX = 300 \text{ USD} \tag{2.18}$$
$$MaxCost = 1200 \text{ USD} \tag{2.19}$$

The validity of the link loads are evaluated:

$$\begin{array}{lcccccl} L^1_{i\neq j}\,\forall l_{ij} > 0 & = & [1600,700,300,400] & \leq & LinkCap & \rightarrow & \text{Link loads valid!} \\ L^2_{i\neq j}\,\forall l_{ij} > 0 & = & [700,300,600] & \leq & LinkCap & \rightarrow & \text{Link loads valid!} \end{array} \tag{2.20}$$

the validity of the cache loads are evaluated:

$$\begin{array}{lcccccl} L^1_{i=j}\,\forall l_{ij} > 0 & = & [2600] & \leq & CacheCap & \rightarrow & \text{Cache loads } \textbf{exceeded}! \\ L^2_{i=j}\,\forall l_{ij} > 0 & = & [1600,1000] & \leq & CacheCap & \rightarrow & \text{Cache loads valid!} \end{array} \tag{2.21}$$

and the validity of the costs are evaluated:

$$\begin{array}{lcccccccl} Cost(S_1) & = & 1\cdot(200+300) & = & 500 & \leq & MaxCost & \rightarrow & \text{Cost valid!} \\ Cost(S_2) & = & 2\cdot(200+300) & = & 1000 & \leq & MaxCost & \rightarrow & \text{Cost valid!} \end{array} \tag{2.22}$$

Since solution 1, has an exceeded cache load it is an infeasible solution, meaning that solution 1 is the best choice. If solution 1 had a valid cache load the choice of the best solution is performed by:

$$\min(l^1_{max}, l^2_{max}) = \min(1600, 700) = 700 = l^2_{max} \tag{2.23}$$

2.3 Scenarios

In order to generate realistic network loads from a cache deployment, realistic distributions of NTs is needed. This project focuses on the Municipality of Bornholm, which is an island in Denmark. The households (NTs) of Bornholm are seen in figure 2.3, while the statistics of the island are seen in table 2.1.

Description	Value
Area	589.38 km^2
Network terminations (NTs)	31999
Central offices (COs)	42

Table 2.1: Bornholm statistics

The problem formulation specifies the problem of planning the amount and placement of caches in a ISP's network. An ISP's network is represented by planning a deployment of COs with inter-connections using either a ring or double ring (3-connected) topology. It is assumed that the ISP has a simple one layer network architecture, where central offices are physically inter-connected using a specific topology. This allows for results that can be analysed for network architectures of low complexity. As the task of planning a deployment of COs is not the focus of this project, NTs, CO locations and their inter-connection are provided by Jose Manuel Gutierrez Lopez (Post doc, Department of Electronic Systems, Aalborg University). The locations of COs is seen in figure 2.4, where each NT is associated to a CO. Two scenarios are described in the following sections, by a type of topology. The ring and double ring topologies are chosen, because the results are simple to analyse, while also allowing for a comparison of results for topologies with different connectivity.

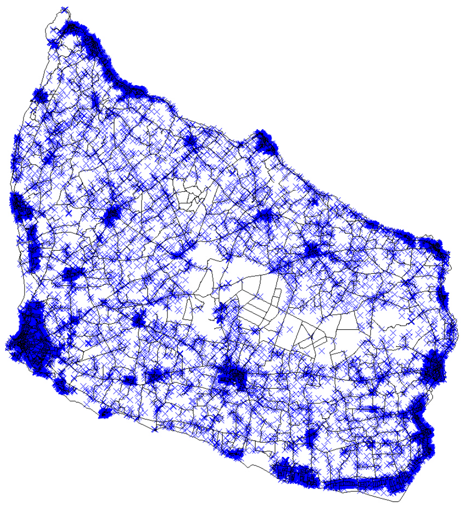


Figure 2.3: The island of Bornholm, Denmark.
Blue crosses represent households (NTs), while black lines represent the road system (segment points)

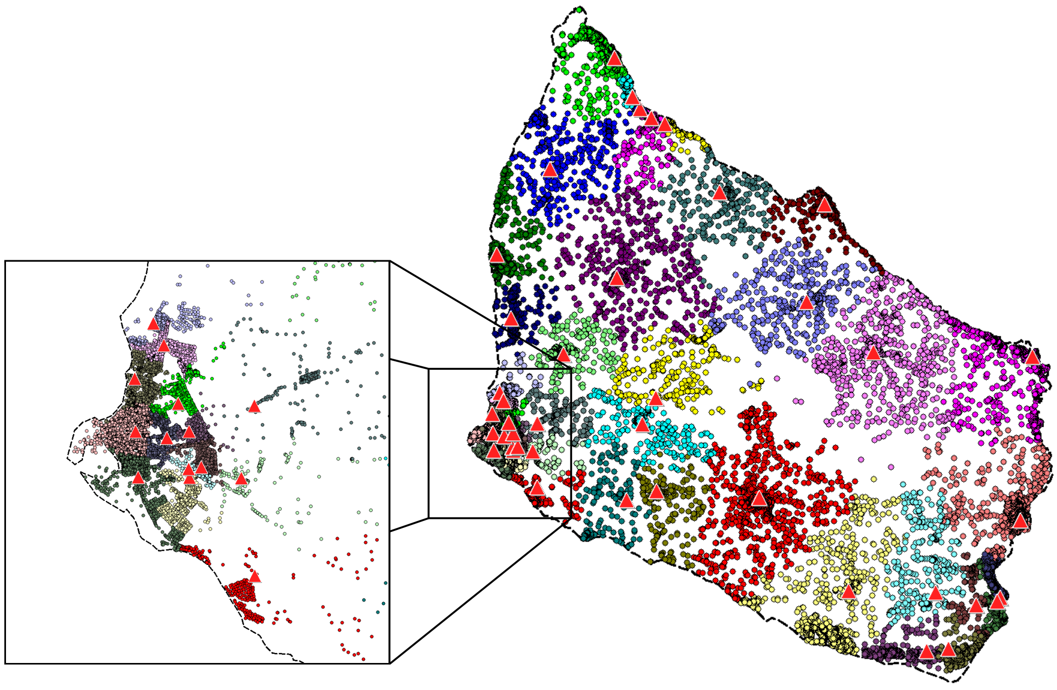


Figure 2.4: COs (triangles) and their associated NTs (circles) on Bornholm. The different colouring shows the associations of NTs to a CO. An area is magnified in order to separate the areas with different CO associations.

2.3.1 Ring

The ring topology is one of the most simple network topologies. The ring topology is a 2-connected graph, which means that a removal of 1 CO or connection will not disconnect the graph. The general definition is that for a k-connected graph, any removal of a subset of $k-1$ vertices or edges will still result in a connected graph. The ring topology applied to the deployment of COs is seen in figure 2.5. The NT population associated to each CO is seen in the bar plot in figure 2.6. The bar plot clearly shows a peaky nature, which is an indicator of a large population in the area where the CO is located. The populations can be inspected graphically in figure 2.4, where the NT to CO associations are seen.

It is expected that the resulting placement of caches will have some correlation to the peaks in NT populations, because these peaks are significantly larger than the average NT population of approximately 762 NTs. By placing caches at these locations, the COs' contribution to link loads are nullified when assuming that the whole NT population from a CO is associated to one cache. The nullification of link load is due to the CO's NT population is due to the NT population being proportional to the amount of traffic originating from the CO.

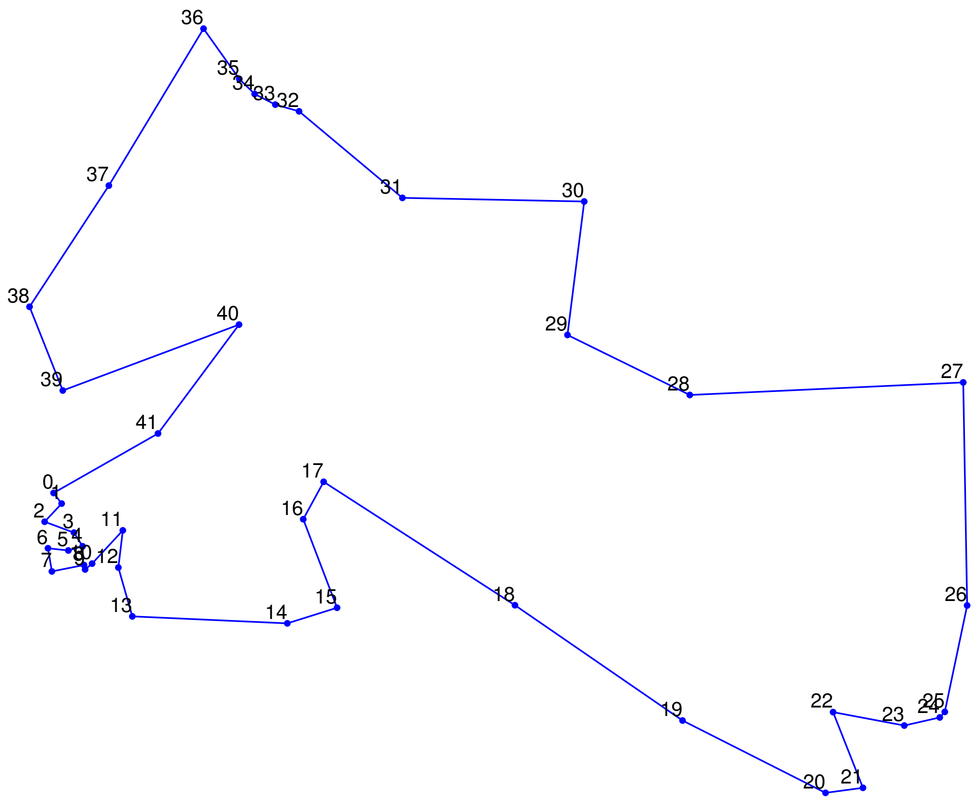


Figure 2.5: COs interconnected using ring topology

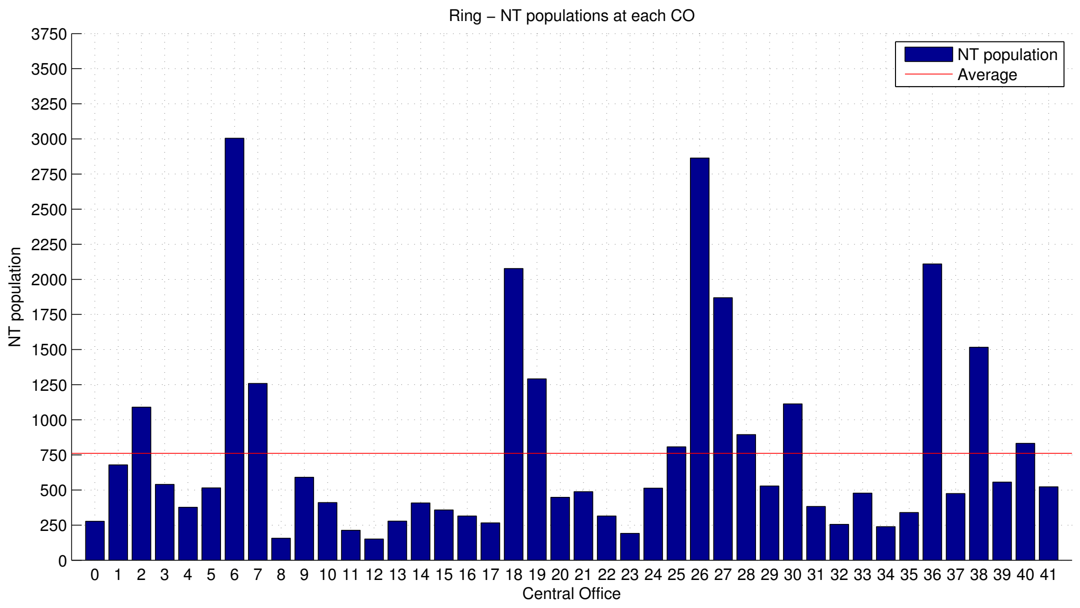


Figure 2.6: Bar plot of NT population at each CO in the ring topology.

2.3.2 Double ring

A 3-connected double ring is used as the second scenario. The graph being 3-connected, implies that a removal of 2 COs or connections will not disconnect the graph. The topology is applied to the CO deployment in figure 2.7 and the NT populations for each CO is seen in figure 2.8. Although the NT population at each CO are identical to those from the ring topology, their numbering is different. This specific numbering allows one to associate the numbering to the two rings, where CO 0-20 are in ring 1 and 21-41 are in ring 2. Each ring contains 21 COs. The two rings are interconnected using $X \leftrightarrow X + N/2$, where CO X is connected to CO $\{X + N/2\}$ and $N = 42$ is the number of COs. This means that CO 0 connects to CO 21, 1 to 22, 2 to 23, ... , and 20 to 41.

Comparing the NT populations of the two rings, it is seen that one ring will sometimes dominate in the amount of NTs associated to it when considering the inter-connections along the two rings. For example, in ring 1, 5-10, and ring 2, 26-31, ring 2 has a much larger percentage of the NTs in the area. This is definitely not always the case, but is an interesting observation from the provided NT to CO association. It is like the ring topology, expected that the caches be placed at locations with large NT populations, but the resulting cache deployment should not be the same. This is due to the fact that the connectivity is higher for a double ring, allowing for a larger amount of possible paths. This should result in a decrease in link loads, as it allows for a more even distribution of how many COs that occupy a link.

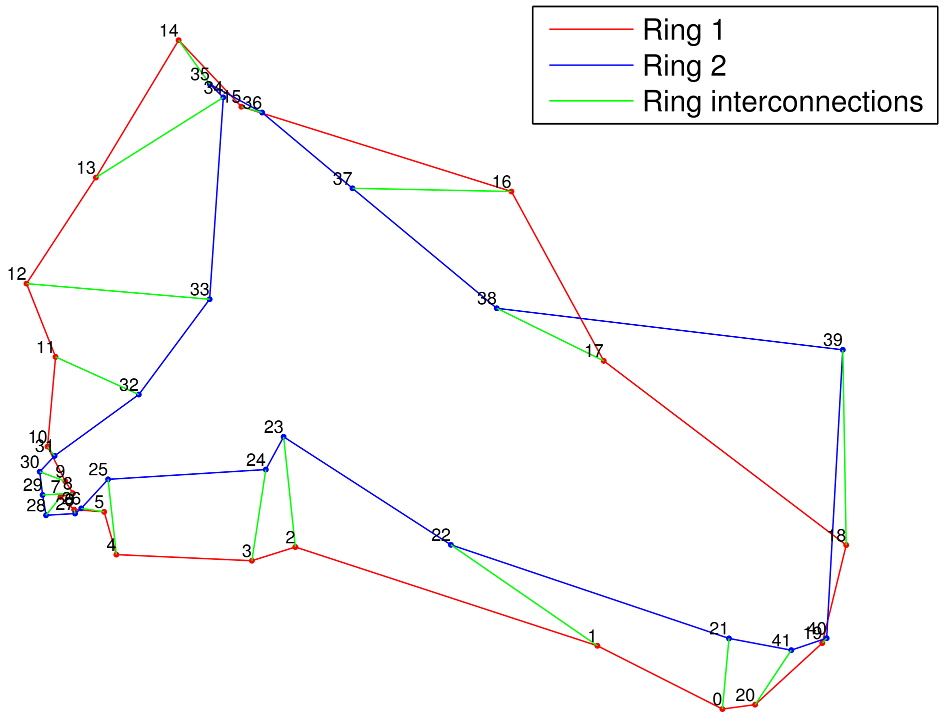


Figure 2.7: COs inter-connected using double ring topology

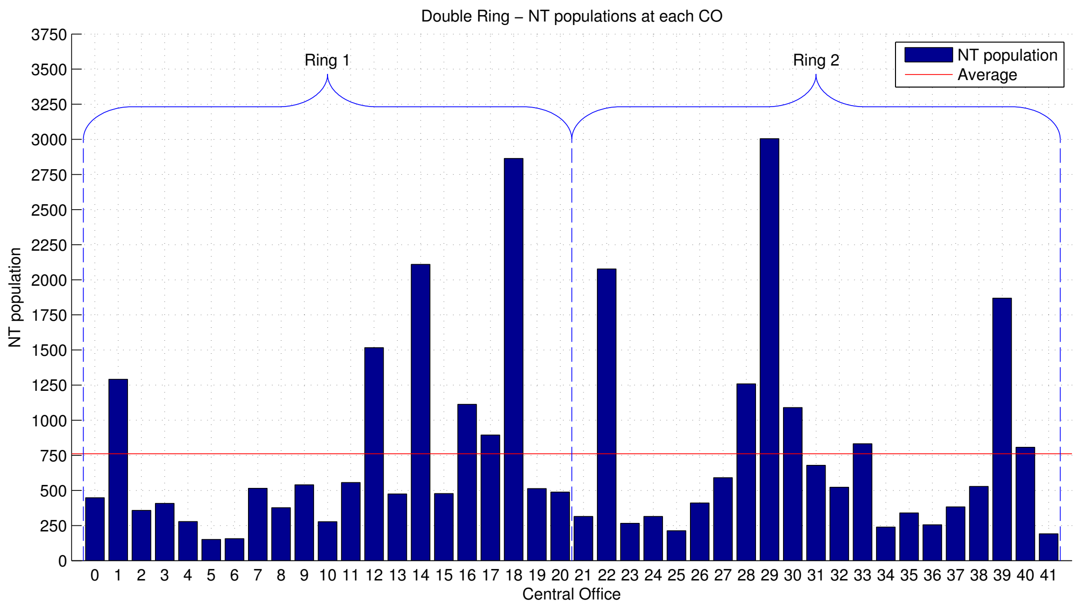


Figure 2.8: Bar plot of NT population at each CO in the double ring topology.

2.3.3 Comparing the two topologies

The two presented scenarios have different characteristics, mainly in the number of edges and connectivity. By increasing the connectivity of the COs in the network, a decrease in network diameter and average path length is achieved. The network diameter is defined by the maximum shortest path between any CO. The average path length is likewise derived from shortest paths. By applying the search algorithm to find a good deployment of caches, it will be interesting to see how the differences in the topology affects the results of the algorithm. A smaller network load is expected for the double ring topology, since it contains more edges and therefore more paths. An increase in quality of service is also expected, since the average path length is decreased, resulting in delays that on average are smaller. All distances are calculated assuming line of sight distances corresponding to those seen in figures 2.5 and 2.7. Minimum and maximum distances are derived from the distances between any CO that is inter-connected. This shows that the double ring topology will result in an increase in maximum distance, due to an increase in connectivity.

	Ring	Double ring
Nodes	42	42
Edges	42	63
Connectivity	2	3
Trenching [km]	124.04	616.16
Min. distance [km]	0.17	0.17
Max. distance [km]	8.51	28.25
Diameter [km]	61.96	60.48
Avg. path length [km]	29.74	24.83

Table 2.2: Statistics for the two topologies

Using the two presented scenarios as input, the local search algorithm is applied and a solution is found for a deployment of caches. The information given in this section can easily be mapped to the content of the generic definition in section 2.1.4. The provided COs are placed in the set of COs, V, the two topologies are represented using the adjacency matrix, A, the NT population is represented using the NT population vector, N, and the link cost matrix, C, contains the euclidean distances between each CO with an adjacency.

2.4 Conclusion

This section summarises the main points of the problem formulation. The focus of the problem formulation is planning the amount and placement of caches for OTT content in an ISP's network that provides a sufficiently load balanced backhaul and is good at avoiding congestion. The problem is a facility location problem and is defined as the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem. An optimisation algorithm is used to find a deployment of caches that yields the best balancing of the backhaul network. The optimisation algorithm should searches for a deployment of caches with a minimal l_{max}. The input to the algorithm is an undirected input graph G_I, containing information on COs, connectivity, NT populations and link costs. The output of the algorithm is a directed solution graph G_S, containing G_I, open caches, paths from COs to caches, link/cache loads and a cost of the solution. A solution is feasible if constraints to the cost of the solution is upheld. The cost can easily be mapped to a number of caches instead of a currency. Two scenarios are given with the same CO deployment but with two different topologies applied; ring and double ring. Having these two topologies will show how the results change when using different types of topologies.

Now that the problem formulation and the scenarios have been presented, an analysis is presented in the following chapter providing a better picture of the different aspects that are a part of this master thesis.

Chapter 3

Analysis

This chapter contains an analysis of ...

3.1 Solving the LLB-CFL problem

In order to find a deployment of caches that provide a good offload of an ISP's backhaul, the field of combinatorial optimisation is analysed. Combinatorial optimisation consists of finding an optimal solution from a finite set of solutions. In many problems within combinatorial optimisation problems, exhaustive search will have very long running times, due to a large solution space. This is also the case for the problem of placing caches. In equation 3.1 the calculation for how many different possibilities there are of placing k caches in a network with n CO locations. It is calculated from the principle of finding k-combinations through the binomial coefficient.

$$N_p(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3.1}$$

A calculation of an exhaustive algorithm's execution time can be performed, by applying the problem of 42 potential CO locations and assuming a symbolically short computation time of 10 milliseconds from routing the demands from COs to caches to finding link and cache loads. A running time calculation is seen in equation 3.2 and is depicted in figure 3.1.

$$T_p = 10\text{ ms} \cdot N_p(42,k) \tag{3.2}$$

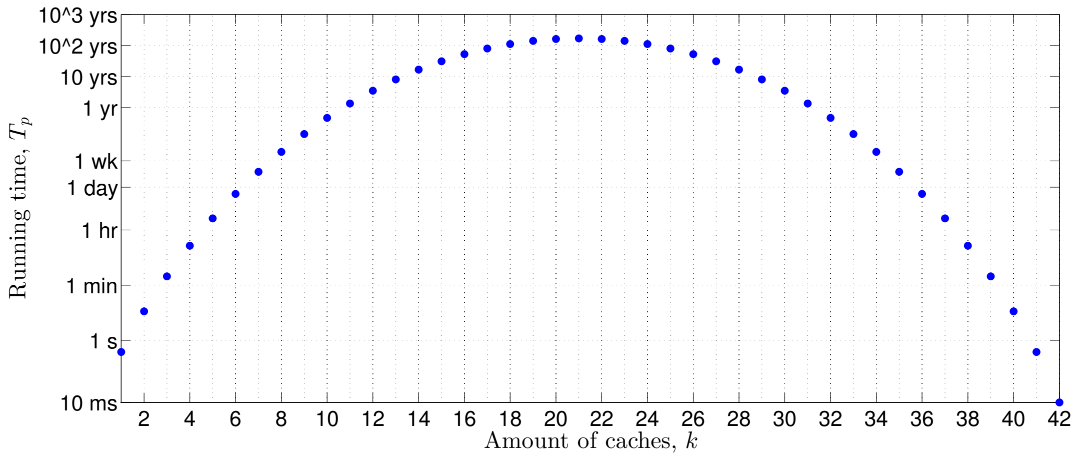


Figure 3.1: Exhaustive search running time for planning of different amounts of caches. The y-axis is logarithmic with ticks at relevant time units.

From figure 3.1 of the running time over amount of caches, the complexity of the problem changes for different amounts of caches. It is seen that in the interval of 7-35 caches the running time will start taking over a day and already at 11-31 caches it will take over a year. It is evident that an algorithm is required, which can find an sufficiently good solution without requiring a search of the entire solution space.

To calculate a trade-off between the cost of a solution and how well the ISP's backhaul is offloaded, an interval of the amount of caches can be calculated. By calculating a solution for each amount of caches, starting at the smallest, a analysis of the trade-off is performed. If a specific maximum link load is sought after, the search for it will start at the same interval and stop when a solution is found for a specific amount of caches. The computation time involved in calculating such a task, will be summed across the interval investigated.

A handful of combinatorial optimisation algorithms have successfully been used to solve general facility location problems using exact, heuristic and metaheuristic algorithms. The facility location problems and all its variations are known to be NP-hard combinatorial optimisation problems [Li and Yeh, 2005, p. 581]. An exact algorithm such as branch-and-bound (linear programming) will solve the problem and find a global optimum solution, but is time consuming to a degree that is impractical [Li and Yeh, 2005, p. 128][Gong et al., 1997]. Therefore the field of facility location has exhibited usage of heuristic and metaheuristic algorithms to find local optimum solutions within a reasonable amount of time. Heuristic algorithms have the limitation that they can terminate at local optimum, while metaheuristic algorithms implements methods for avoiding termination at bad local optimum solutions [Gong et al., 1997]. Metaheuristic methods are therefore good at solving facility location problems. Metaheuristic methods are among others tabu search, simulated annealing, Genetic Algorithm (GA), variable neighborhood search and ant systems, which all try to avoid terminating at local optimum solutions [Li and Yeh, 2005, p. 105].

Even though this problem can be solved with any of the mentioned metaheurstic algorithms, GA is chosen, as they have shown good results in the field of facility location and are conceptually simple, broadly applicable and can provide good local optimal solutions within feasible running times [Li and Yeh, 2005]. The idea is that by providing a fitness function that defines how well a solution is at solving the problem at hand the GA can search for a local optimal solution through natural evolution. By iteratively applying reproduction operators, inspired by natural evolution, on a set of solutions, a local optimal solution is found. The broad applicability allows a GA to be applied to optimisation problem, by simply exchanging the fitness function. In the case of this thesis, an implementation of a routing algorithm is easily interchangeable.

3.2 Genetic Algorithm

This sections includes an introduction of the genetic algorithm and various techniques that can be used. Only the most popular and those that are applicable to the problem are presented, as there are many techniques. Many of these techniques may work differently based on the problem at hand, but nevertheless advantages, disadvantages and applicability to the problem are analysed. Knowledge from this section is inspired from Sivanandam and Deepa [2007], as they have written a good introduction to GAs, including not only a large amount of techniques but also thoughts into how various techniques affect the algorithm and its applications. Techniques used in state of the art are also presented, where it is clearly indicated when they are presented.

3.2.1 Background

GAs are a popular approach to solving complex combinatorial optimisation problems. Its principles were first introduced in 1973 by Holland. These include the principles of genetics and natural evolution and how to apply them to search and optimisation problems. GA is a metaheuristic (non-problem specific heuristic) that performs a guided random search through a solution space, and provides local optimum solutions to a wide range of problems.

In a GA, a representation scheme is chosen that defines the set of solutions corresponding to a search space. A number of individual solutions are generated to form an initial fixed-size population. The fitness of each of these individuals are evaluated through a fitness function that is specifically designed to solve the problem at hand. A number of individuals are chosen based on their fitness to be parents. New individuals are produced from these parents using reproduction operators. The population of the next generation is selected from the offspring of reproduction and the old population. A selection method defines which parents must reproduce and which individuals that survive into the next generation. When a termination criterion indicate that an inescapable local optimal solution is found among the population, the algorithm terminates. The following sections describe the techniques that can be applied to the described aspects of GAs. The sections also relate the aspects of GAs to the formal definition (see section 2.1.4).

3.2.2 Applications in literature

Although no research can be found that relate to solve the facility location problem based on load balancing, some inspiration can be achieved from looking into how others have solved facility location problems with genetic algorithms. Facility location problems primarily focus on minimising distances or delays. One facility location problem that is somewhat related to this project is the capacitated p-median problem, which is solved using GA by Correa et al. [2001]. The capacitated p-median problem consists of locating p facilities which satisfy n demand points, such that the total sum of distances travelled, from each demand point to its nearest facility (median), is minimised. Each facility has a fixed capacity, which can only service a specific amount of demand. These problems are NP-hard. The capacitated p-median problem solved by Correa et al. places 26 exam locations given 43 potential location to minimise the summed distance travelled for each individual student (19710 total). They perform custom reproduction operators which allow the genetic algorithm to be more problem specific by only producing feasible solutions corresponding to the amount of exam locations. This avoid searching a infeasible part of the solution space.

Li and Yeh [2005] propose a genetic algorithm that uses GIS data such as population coverage, transportation cost and road proximity to solve the facility location problem of placing a different amounts of hospitals in a grid of size $300m^2$ and dimensions 150×150 (22500 points). The GIS data is used to perform multi-objective optimisation by maximising the population coverage, minimising transportation cost and minimising proximity to roads. This was performed by normalising the individual fitnesses so they have equal weight.

Another application of genetic algorithms in facility location is performed by Sasaki et al. [2010], where an optimal location of 2 to 50 ambulances from 175 potential locations is found. The demand at a potential location is given by the amount of emergency cases associated to it. The fitness function minimises the summed distance, where the individual distance from each potential location is weighted by the amount of emergency cases associated to it. Using future predictions to emergency cases (2030) and actual (2007) emergency cases, future ambulance locations and improvements to current ambulance locations are proposed.

3.2.3 Description

This section presents the GA and its main aspects. It will also include various methods for applying genetic algorithms that are relevant for the problem of this thesis.

Representation

An individual represents a solution. These individuals are in terms of genetics, called chromosomes. When applied binary encoding, a chromosome can be represented as e.g. the bit string [1 0 1 1 0 1]. Chromosomes constitute a sequence of genes. This does not necessarily mean that each bit in a bit encoded chromosome represents a gene, as a gene can represent multiple bits. The example chromosome could contain 3 or 6 genes as seen in table 3.1.

3 genes	1 0	1 1	0 1

6 genes	1	0	1	1	0	1

Table 3.1: Example of equal sized chromosomes with different gene sizes

Although most common, binary encoding is not the only way genes can be represented in a chromosome. A couple of other possibilities are octal, hexidecimal, permutation (real numbers from sequence) and value (e.g. 123, 'A', 'black') encoding.

A chromosome consists of genes. A phenotype is the chromosome expressed in the domain of the model. A morphogenesis function maps the chromosome to the phenotype. The representation explained in the formal definition in the problem formulation (see section 2.1.4) is a binary encoded chromosome with one gene corresponding to a single bit. The morphogenesis function then associates the index of a bit in the bit string to a location in a specific network and associates its bit value to represent if a cache is deployed at that location (bit index).

Fitness

A fitness indicates how good a chromosome is at solving the specific problem and how close the chromosome is to an optimum. The fitness of a solution is a value calculated from a fitness function using a phenotype, decoded from a chromosome. The fitness function is problem specific, as it evaluates the phenotype representation of the solution. For multicriterion optimization, the fitness may be more complex to determine. In this case there is a dilemma in defining the differences between the solutions and which is better. If it is possible to combine these criteria it must be consistent. One must also have thoughts on how much each of these criteria contribute in finding the optimum when combining them. For other cases where it is not simple to combine criteria, pareto optimality can be considered. Consistency of fitness function evaluation is also required for the single criterion case, meaning it must be non-random and be non-noisy.

The fitness evaluation for this thesis' problem involves routing traffic from COs to caches and finding the link loads in the network, as is explained in the formal definition (see section 2.1.4). The fitness value is then given by the maximum link load of the network, which represents how load balanced the network is.

Population

The population is populated with individuals and has two important aspect for GAs that can be manipulated. These are generation of the initial population and population size. An example initial population is seen in table 3.2, where the population size is 4.

Chromosome 1	0	1	0	0	1	1
Chromosome 2	1	0	1	0	1	0
Chromosome 3	1	1	0	0	0	1
Chromosome 4	1	0	1	1	1	0

Table 3.2: Example initial population

The initial population should have a large gene pool (large diversity) allowing for the algorithm to initially search the whole search space. This is usually performed through random initialisation. It may be preferable to apply some heuristic to initialise the population with a better mean fitness, which can reduce computation time. But, in these cases there must be a large enough gene pool to search more than just a small part of the solution space.

By increasing the population size, it becomes easier to explore the search space. It will on the other hand increase computation time. A convergence has occurred if the individuals are alike and only a mutation will result in finding a better solution. It has also been shown that the population size is a determining factor for when global and local optimum solutions are found. In the end a bigger population size is desirable, but it comes at the expense of larger computational cost, memory and time. Sivanandam and Deepa [2007] write that a population size of 100 individuals is the most popular, but the size should be adapted to the problem and and that there is a trade-off between computation time and quality of result.

Genetic algorithm process

A GA consists of iterative search cycles consisting of the following steps; Initialising the population, evaluating each individual's fitness, reproduction for the next generation and decision of termination according to some criterion. Reproduction is a central part of GAs. Reproduction operators guide the search process by producing new and possibly fitter individuals. These operators consist of:

- **Selecting** a number of individuals from the population to be parents
- Performing **Crossover** on the selected parents to create new individuals also known as offspring or children
- **Mutating** a number of individuals to avoid local minima traps through diversity.
- **Replacement** of individuals with the newly produced individuals to create the next generation of individuals.

The process of a GA is seen in figure 3.2 as a flow chart, where the mentioned aspects are depicted.

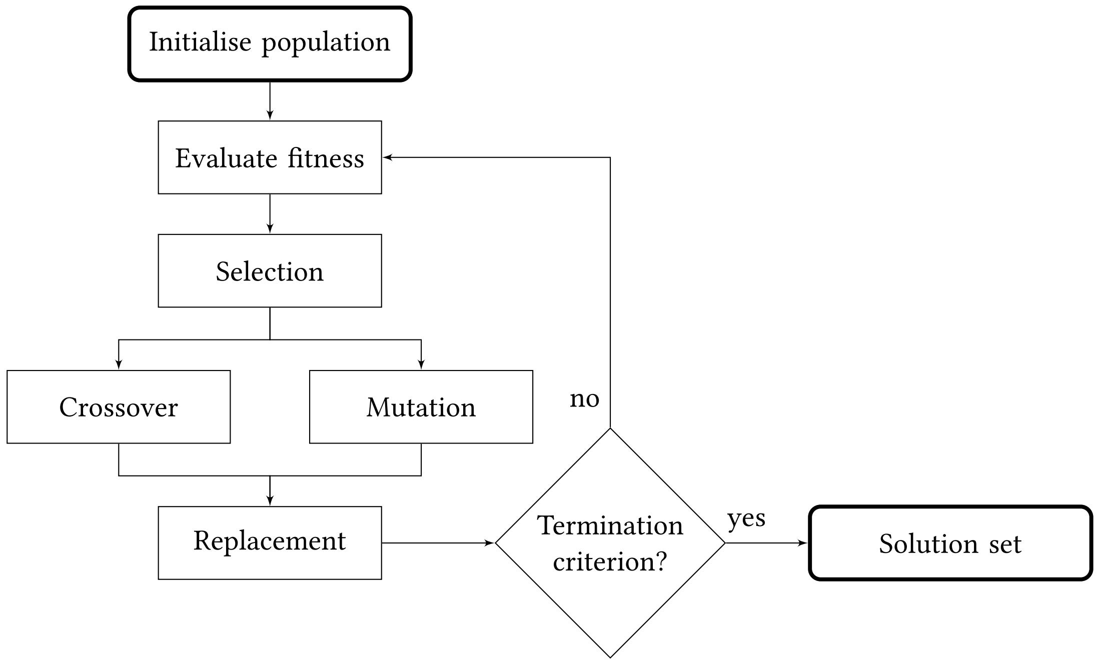


Figure 3.2: Flow chart of a Genetic Algorithm (GA) process [Sivanandam and Deepa, 2007, pp. 30-31]

Different aspects of the GA are presented in the following including a thorough explanation of each reproduction operator. As there are many different methods, only the most popular are presented.

Selection

The task of the selection process is to select individuals from the population that will reproduce and create offspring for the next generation. The selection method should generally select individuals with higher fitness in the hopes that their offspring will gain even higher fitness. Selection is a random process in that it randomly selects individuals dependent on their fitness value. A selection pressure defines how favoured high fitness individuals are in the selection method. A higher pressure results in higher fitness individuals being selected more often. The convergence of GAs is influenced by the selection pressure. A higher selection pressure results in a higher convergence rate. A high selection pressure can result in premature convergence, resulting in sub-optimal solutions. If the selection pressure is low, the convergence rate is slow, meaning that the GA will take a larger amount of computation time to find a solution. It is the diversity in the population that avoids premature convergence, so a trade-off is made between diversity and convergence rate when choosing the selection pressure.

Selection methods can be distinguished by two types of selection, proportionate-based selection and ordinal-based selection. The proportionate-based selection methods select individuals with a probability proportionate to the size of their fitness. Ordinal-based selection methods select individuals based on the ranking of the individual, through ordering of fitness values in the population.

Fitness scaling can be applied to fitness values to improve the selection process [Matlab, 2014a]. A higher probability of selection is assigned to individuals with a higher fitness value. The range of these values also affect the probability of selection. If the variance in the fitness values is too high, the individuals with the highest fitness will have a too high probability of selection resulting in a search of the solution space using mostly high fitness individuals. If the values have a too small variance, the probability of selection will be identical resulting in a very slow convergence.

Bias and spread are also important factors to consider when choosing a selection method [GEATbx, 2014]. The bias describes the absolute difference between the actual and expected probability of an individual being selected. Zero bias achieves an actual probability equal to the expected probability of selection. It is necessary to have zero bias, if a selection pressure is to be applied. A spread defines a range of possible times a individual can be selected for reproduction from the same population. Minimum spread is the smallest spread that achieves zero bias. Minimum spread allows for a guarantee for the minimum amount of times that individual is selected for reproduction.

Various methods for selection are presented in the following. This includes methods that directly affect the selection process specifically proportionate selection and ordinal selection. Fitness scaling, for improving the selection process, is also presented.

Proportionate selection

Proportionate selection includes methods such as roulette wheel and stochastic universal selection [Sastry et al., 2005, pp. 99-100]. For the roulette wheel selection method, an individual is assigned to a slot in a roulette wheel with a size proportionate to its fitness. The larger the fitness the larger the slot size. By *spinning* the roulette wheel, a reproduction candidate is selected. A roulette wheel implementation consists of the following points (as described by Sastry et al.):

1. Each individual's fitness, f_i, is evaluated.
2. Propability (slot size), π_i, of selecting an individual as a reproduction candidate is calculated as seen in equation 3.3, where $i = 1..n$ and n is the population size.

$$\pi_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{3.3}$$

3. The cumulative probability, q_i, is calculated for each individual as seen in equation 3.4.

$$q_i = \sum_{j=1}^{i} p_j \tag{3.4}$$

4. A uniform random number, $r \in (0,1]$, is generated.
5. In the case where $r < q_1$, select the first individual. If this is not the case select the individual where $q_i - 1 < r \leq q_i$.
6. Steps 4-5 are repeated n times to select n reproduction candidates.

The roulette wheel selection method has zero bias, but does not guarantee a minimum spread. Selective pressure is not a parameter for roulette wheel selection, as it is determined by the selection probabilities that are dependent on the fitnesses. To apply a certain selection pressure fitness scaling is required. In certain cases where an individual occupies e.g. 90% of the probability due to a very high fitness, the probability of choosing low fitnesses will be very small. This indicates a too high selective pressure and will result in premature convergence.

To illustrate the roulette wheel method, an example of the roulette wheel selection method, a population of $n = 6$ individuals is seen in table 3.3 and figure 3.4. The total fitness is $\sum_{i=1}^{n} f_i = 10+5+20+40+30+15 = 120$. The total fitness is used to calculate the respective probabilities, π_i. The cumulative probability are also shown. If a random uniform number e.g. $r = 0.3$ is generated, then individual 4 is selected, since $(q_3 = 0.29) < (r = 0.3) \leq (q_4 = 0.63)$.

Individual	1	2	3	4	5	6
Fitness, f_i	10	5	20	40	30	15
Probability, π_i	10/120≈0.08	0.04	0.17	0.33	0.25	0.13
Cumulative probability, q_i	0.08	0.12	0.29	0.62	0.87	1.00

Table 3.3: Statistics for example of roulette wheel selection method

Figure 3.3 shows the process of selecting $n = 6$ new individuals. It is clearly seen here that there is no guarantee to the minimum spread of the amount of times an individual is selected for reproduction, since a uniformly generated variable that generates point of selection can give no guarantee to spread between the variables.

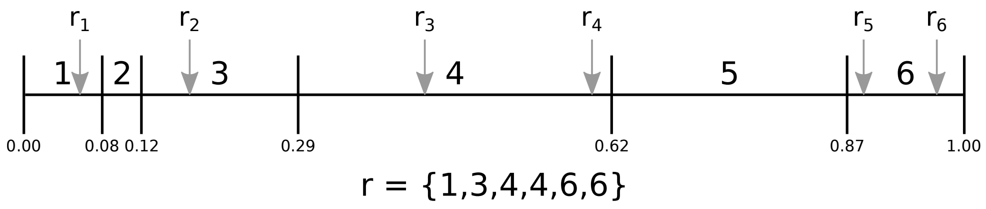


Figure 3.3: Example of roulette wheel selection process

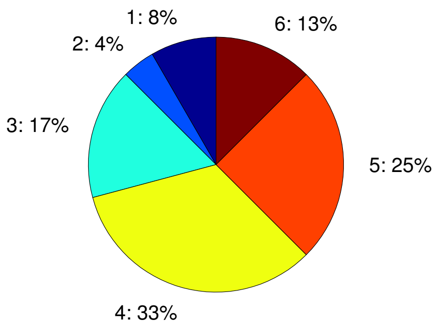


Figure 3.4: Graphical example of probabilities in a roulette wheel

Selection by stochastic universal sampling achieves both zero bias and minimum spread, unlike roulette wheel selection. Individuals are mapped onto a line alike the roulette wheel as seen in figure 3.3. But only a single random number is generated to select any number of individuals for reproduction. The first random number r_1 is generated in the range $[0, NPointer]$, where $NPointer$ is the number of individuals selected. The remaining numbers $r_2 - r_6$ are seperated with an equal distance of $1/NPointer$. If $n = 6 = NPointer$ the distance is $1/6 \approx 0.167$. It is this equal distance, that allows for a gurantee to the minimum spread. This selection method also guarantees that both high fitness and low fitness individuals will be part of the selection for each generation. Figure 3.5 shows the stochastic universal sampling process.

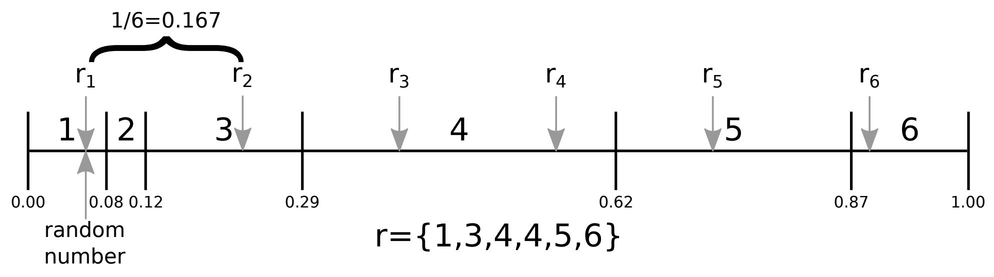


Figure 3.5: Example of stochastic universal sampling selection process

Ordinal selection
Methods such as tournament selection and truncation selection constitute ordinal selection [Sastry et al., 2005, p.100]. Tournament selection is performed by choosing s individuals at random, with or without replacement. These individuals enter into a tournament where the fittest individual is selected. This is performed until n individuals are selected through n tournaments. For truncation selection, the selection is performed by selecting the $1/s$ fittest individuals and replicating them each s times.

With proportionate selection there is a constant selective pressure applied. An increase in selective pressure for tournament selection is performed by increasing the amount of individuals part of each tournament, s. Likewise, selective pressure is increased in truncation selection by increasing s.

Fitness scaling
For a method such as roulette wheel selection, the probability of selection is highly dependent on the fitness value. If the variance is high, low fitness individuals will have a very low probability of selection, which may not be desirable as this results in low diversity. Rank fitness scaling removes this effect [Matlab, 2014a]. This is done by sorting according to the fitnesses and assigning an individual with rank n with a scaled fitness according to $1/\sqrt{n}$. This means that the lowest value fitness maps to rank 1. The sum of the scaled values must equal the number of parents required for reproduction, which is performed by $value(i) = n_{parents} \cdot \frac{value(i)}{\sum(value)}$. Figure 3.6 shows an example of fitness rank scaling, where $n_{parents} = 6$.

This method of rank scaling pertains to non-linear fitness rank scaling dependent on required parents. Linear rank scaling is also possible, where a fitness is mapped to a function that is linear. The selection pressure is performed by e.g. adjusting the function of the non-linear ranking to alter the probability distribution. Likewise can be done for the linear rank function by adjusting its slope.

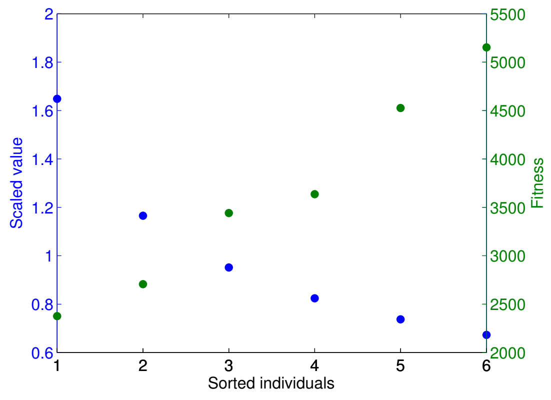


Figure 3.6: Example of fitness rank scaling

Crossover

Crossover is a reproduction operator that produces an offspring from two parents. After selection has been performed and a mating pool has been created, a crossover is applied to pairs from the mating pool. The hope is that performing a crossover on these parents will result in offspring with a better fitness. A crossover probability determines the amount of offspring produced using crossover. A crossover probability of 100% indicates that all offspring are produced through crossover. Opposite, a crossover probability of 0% indicates that no offspring is produced through crossover and the population is unaltered.

Different crossover functions are presented in the following relating to the representation method of this thesis, namely bit string.

One point crossover

The most simple crossover is one point crossover. A crossover point is generated using a uniform random number generator. The section of the bit string after this crossover point is exchanged between the two parents to create offspring. Figure 3.7 depicts this type of crossover.

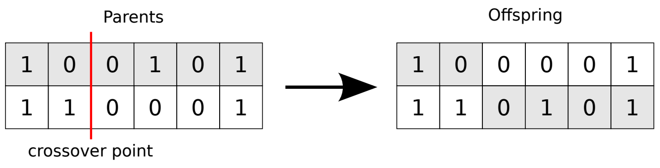


Figure 3.7: Example of one point crossover

Two point crossover

Two point crossover allows an exchange of a section of bits within a range using two crossover points. The bits between these two crossover points are exchanged between the parents to create new offspring. Figure 3.8 shows how a two point crossover is performed.

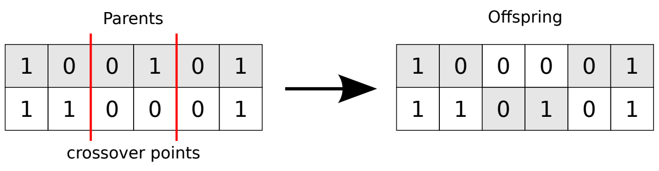


Figure 3.8: Example of two point crossover

The reason for using higher amounts of crossover points is because with one point crossover there is no way for offspring to keep the good fitness attained from a parent with good genetic information in both head and tail bit of one parent. This is what 2 point crossover allows. This problem is not only evident for the head and tail bit, but for each bit in the bit string. Beause of this, two point crossover will allow for more complete search of the solution space compared to one point crossover.

N point crossover

N point crossover is alike two point crossover, but with allows for additional sections to be exchanged. Sections for crossover are chosen in even numbers as seen in figure 3.9. There exists 4 sections in the example. This means that crossover is performed on sections 2 and 4.

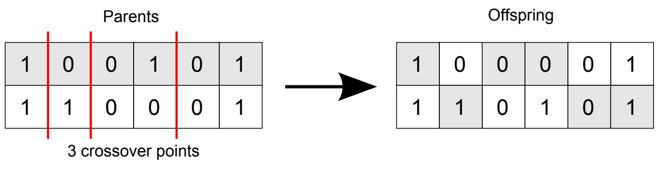


Figure 3.9: Example of N point crossover

Uniform crossover

Uniform crossover is performed by performing random uniform crossover between genes instead of deciding on a crossover point. A uniformly generated vector decides what parent the offspring will inherit from. For the first child, it will inherit from parent 1 if the vector is a 1, and from parent 2 if the vector is a 0. The opposite applies to child 2. This type of crossover is seen in figure 3.10.

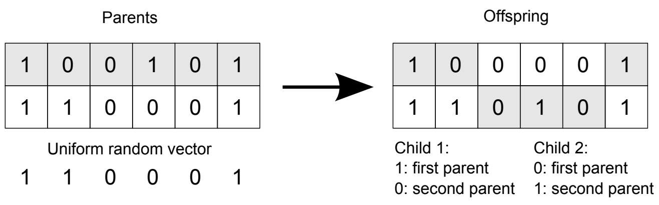


Figure 3.10: Example of uniform crossover

Exchange crossover

Due to the problem formulation of this project the generic crossover methods are not applicable. This is due to the fact that none of the mentioned crossover methods guarantee that a exact cost/amount of caches is adhered to. The cost is proportional to the amount of caches deployed. When assuming no capital expenditure (CAPEX) or operational expenditure (OPEX) and cost of caches equal to one, the cost is equal to the amount of caches. The cost is as seen in equation 3.5, a sum of the bit string S. This means that when a crossover is performed, and no infeasible solutions are allowed to be created, the sum of bits in the bit string must equal $Cost(S)$.

$$Cost(S) = \sum_{i=1}^{n} (s_i) \tag{3.5}$$

Correa et al. [2001] present a crossover method, where genes are exchanged between parents. By identifying each parents unique genes, these can be exchanged between the two parents to create new offspring. Q_i represents an index vector of where an individual i's genes are 1 and an exchange vector E_i, that represents the difference between the two index vectors. Q_1 and E_1 are associated to the first parent, while Q_2 and E_2 to the second parent. An example of this representation is seen below, while the crossover method is shown in figure 3.11.

$$\begin{array}{rclrcl} Q_1 & = & [0,3,5], & Q_2 & = & [0,1,5] \\ E_1 & = & [3], & E_2 & = & [1] \end{array}$$

This method shows that by exchanging genes, the sum of the bit string for the parents will be the same as with the children, when the parents have the same amount of genes (proper initialisation required). This directly maps to the problem formulation where a specific number of caches are to be deployed. In figure 3.11 one can see that there are 3 genes before and after crossover has been performed.

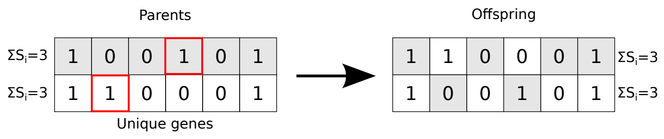


Figure 3.11: Example of exchange crossover from Correa et al. [2001]

Mutation

After crossovers are performed on the population, mutations are performed. Mutation ensures diversity in the population and diversity ensures that local optimum are avoided. This is ensured by preventing the individuals in the population from becoming too similar and thus slowing progress or reaching local optimum. In simpler terms, the crossover operator finds better solutions from those currently in the population, while mutation allows for searching the whole solution space. A mutation probability defines how big a percentage of the chromosome is altered by mutation. A mutation probability of 100% mutates every gene in the chromosome, while 0% mutates no genes in the chromosome. A mutation rate defines how many individuals are mutated. This rate should not be too high, as the GA will turn into a random search.

Different mutation operators are presented in the following relating to the representation method of this thesis, namely bit string.

Flipping
Flipping is performed by choosing an amount of genes in an individual and flipping its bits, meaning a 0 becomes a 1 and a 1 becomes a 0.

Interchanging
Mutation through interchanging bits is performed by choosing two random genes of an individual and interchanging their respective bit values.

Heuristic hypermutation
This operator is proposed by Correa et al. [2001]. This *mutation* is applied with a fixed probability. Algorithm A.1 in appendix A shows the pseudo-code from the algorithm as presented in their paper. The thoughts behind this mutation is to try and improve an existing solution by performing small alterations to a percentage of individuals in the population and continuously evaluating its fitness before performing the mutation. The genes of an individual are altered by moving a deployed facility to each possible location where there is no facility. For each alteration the fitness is evaluated and if there is an improvement the new best individual is saved. To supplement the hypermutation, interchanging is also performed. This is a very heavy operator to perform, as it requires many fitness evaluations. In the genetic algorithm presented by Correa et al., the hypermutation finds a better solution within the same amount of time when being compared to using interchanging by itself.

Replacement

Once reproduction is complete through crossover and mutation the new population must replace the existing one. This creates the new generation of individuals from which the process of selection, crossover and mutation will reiterate. There are two types of replacement possibilities. Generational update or steady state update. Generational update generates offspring for the whole population and then replaces it, while steady state update replaces individuals in the population as soon as new offspring are created.

Elitism allows a number of best individuals to continue to the next generation unchanged. This results in elite individuals contributing to the reproduction in later generations until they are replaced by higher fitness individuals.

Termination criterion

GAs is a search algorithm, which does not know when a global optimum is found, therefore certain termination criterion must be defined to indicate when a solution is good enough or further evolution will not produce better fitness individuals. There known criteria for termination are:

- Maximum generations: The algorithm will stop after a specified number of generations are reached.
- Time limit: After the specified amount of time is reached the algorithm will stop. This is usually specified, if the computation time for the maximum amount of generations is larger than expected.
- Stall generations: If there is no improvement in fitness for a specified number of consecutive stall generations, the algorithm stops.
- Stall time limit: If there is no improvement in fitness for a specified amount of time, the algorithm stops.

3.2.4 Building block hypothesis

The reason why GAs work can be explained through the building block hypothesis. Holland defines a structure named schemas, which describe a subset of genes in an individual. Table 3.4 shows an example of a schema, where $*$ describes a wildcard, while the binary values are fixed. The amount of fixed bits is called the order of the schema. In the case of table 3.4 the order is 2. The hypothesis says that schema resulting in a good fitness will propagate to the next generation. The schema that have low order (few fixed bits) and above average fitness (average fitness from all chromosomes with the specific schema present) are called building blocks. When crossover is performed on individuals with these building blocks, higher order and above average fitness schema can be formed.

Type	Bit string			
Schema	1	*	*	1
Chromosome 1	1	0	0	1
Chromosome 2	1	0	1	1
Chromosome 3	1	1	1	1
Chromosome 4	1	1	0	1

Table 3.4: Example of schema and its possible chromosomes

3.3 Case study

Because there are many different types of caches and many different CDN architectures, a case study of a large OTT CDN is investigated and used to perform assumptions and delimitations in the work of this project. This will e.g. include delimitations to the cache capacities ($CacheCap$) and type of cache. The case study focuses on Netflix, as they produce a large amount of traffic, occupying 31.62% of downstream traffic in North America during peak period. Netflix also openly documents its CDN architecture (unlike the Global Google Cache CDN) and is in the process of defining how large OTT content providers make deals and work together with ISPs to ensure content with sufficient quality [Netflix, 2013b]. Through a description of Netflix's CDN a clear picture can be shown of how OTT content providers plan and manage their network. This information is also used to present how OTT content providers plan a CDN with caches in collaboration with ISP's to ensure streams with sufficient quality at peak hour. The project will, to a high degree, use the specifications of the Netflix CDN to describe how the OTT content of a given network is delivered.

The case study will not only include descriptions of the caches and the CDN workings, but also how an OTT CDN has come to be and what challenges there are associated to building such a network.

3.3.1 Netflix's content delivery network

To give an initial idea of how much downstream traffic is generated due to streaming content from Netflix an estimation is seen in example 3.1.

Example 3.1 (Netflix traffic estimations)
Netflix have streamed 5 billion hours of content in Q3 2013 [Empson, 2013]. Assuming these subscribers all stream with the highest quality stream resulting in a bit rate of 5800 Kbps (Super HD) [Sandvine, 2013, p. 14], the average downstream traffic becomes:

$$\frac{5800 \text{ Kbps} \cdot (5 \cdot 10^9) \text{ hours}}{3 \text{ month}} \approx 12.32 \text{ Tbps} \tag{3.6}$$

This gives a clear picture of how much traffic would traverse the Internet if all Netflix's subscribers were able to stream in Super HD quality.

Super HD quality is not currently achievable for all subscribers of Netflix at peak hour, meaning that the traffic described in example 3.1 in not achievable either. This is due to many different problems relating to how the traffic flows at any given time or disputes to transit costs resulting in congested links, ISPs not adopting Netflix's Open Connect CDN or content not being available in the Super HD. To give a better understanding of these concepts an introduction of how Netflix provides its content is described in the following.

How does Netflix provide its content?

Netflix initially (and still to a small degree) relied on third party CDNs like Akamai, Level 3, Limelight and Amazon [Evers, 2013]. This means that Netflix relied on these third party CDNs to ensure a resilient server infrastructure and ensure good connectivity to its subscribers through peering agreements with ISPs. For an ISP like Level 3, it also meant CAPEX in order to ensure sufficient bandwidth for Netflix [Rayburn, 2010].

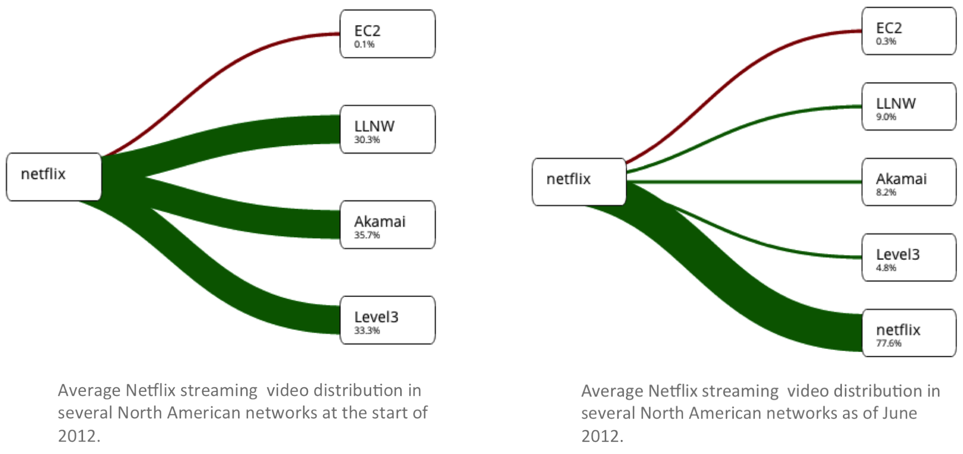

netflix
EC2 0.1%
LLNW 30.3%
Akamai 35.7%
Level3 33.3%
Average Netflix streaming video distribution in several North American networks at the start of 2012.
netflix
EC2 0.3%
LLNW 9.0%
Akamai 8.2%
Level3 4.8%
netflix 77.6%
Average Netflix streaming video distribution in several North American networks as of June 2012.

Figure 3.12: The change in distribution of average Netflix traffic across CDNs from Labovitz [2012], due to the introduction of Netflix Open Connect CDN. The red line is website traffic going to Amazon EC2 (AWS) and green lines are content traffic.

But due to rising traffic demands and a choice to reduce reliance and costs on third party CDNs, Netflix decided to build its own CDN [Rayburn, 2014]. Figure 3.12 shows the distribution of average Netflix traffic across its CDNs and also shows the change in traffic distribution, when Netflix decided to create their own CDN and supply 77.6% of the traffic themselves. Netflix named it Open Connect CDN, and was based on open source software and promoted it to ISPs by saying that "ISPs can directly connect their networks to Open Connect free" [Netflix, 2014]. The idea behind the Netflix Open Connect program is, according to their deployment guide, to "offload the backhaul traffic for ISP's and provide the best user experience to subscribers" [Netflix, 2013b, p. 3].

The Open Connect program consists of partnering with ISPs to place caches (Open Connect Appliances) in their network and to have peering points at major Internet Exchanges (IXs) [Netflix, 2013b, p. 3]. These caches are directed cache appliances, meaning that decisions in regards to which traffic flows to the device is decided by the ISP and Netflix, and not the cache itself. ISPs are given more control over how Netflix traffic flows through their network by mapping these caches to specific IP netblocks (ISP customers). The Netflix Content Delivery System is then able to relay requests for content to the cache that an IP is associated. The cache must of course have the content available and have sufficient capacity to handle the request. These two points are both main contributors when planning for capacity and availability and allows the ISP to control the source and traffic flows in their network. How Netflix's caches work is summarised in the following points:

- One single cache contains the most popular content and offloads approximately 60%-80% of content requests depending on the size of the content catalogue [Netflix, 2013b, p. 5].
- The cache only handles traffic which is directed to it. It does not inspect or intercept traffic in the network. It does not relay requests to other caches either.
- It contains only audio and video files
- A cache serves only subscribers on the IP addresses that it has been associated to.
- Caches are filled nightly with the most popular content for the region.

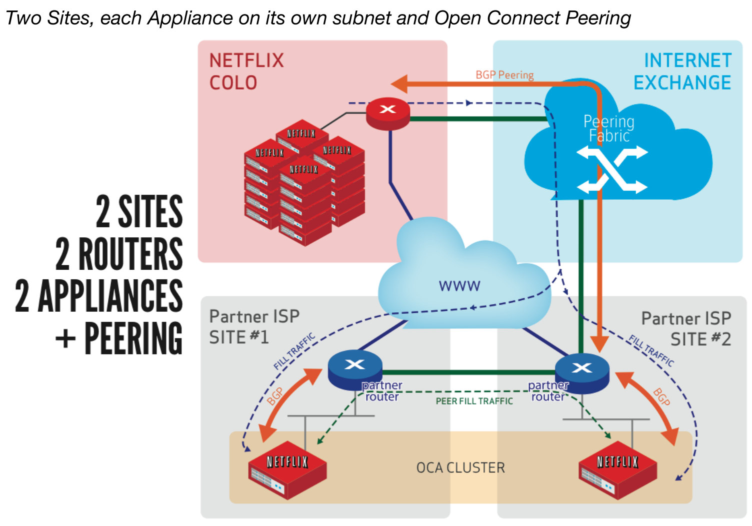


Figure 3.13: Topology of Netflix Open Connect CDN [Netflix, 2013b, p. 30]

The topology of the Netflix Open Connect CDN is seen in figure 3.13. Netflix places servers at co-location centres, where space and bandwidth is rented. Caches (Open Connect Appliances) are deployed in the ISP's network. There are several ways the ISP can use the Netflix Open Connect CDN [Netflix, 2013b, p. 4]. These are:

- Peering with Netflix Open Connect at IXs. The ISP advertises the IP prefixes that should be served via Border Gateway Protocol (BGP).
- Peering with Netflix Open connect at IXs and deploying caches. This is Netflix's typical strategy. Peering will allow for redundancy when caches fail, supply long-tail content and fill traffic (traffic that supplies caches with the most popular content).
- Instead of peering with Netflix, transit networks can be used instead, while still deploying caches. In this case Netflix will not contribute to transit costs.

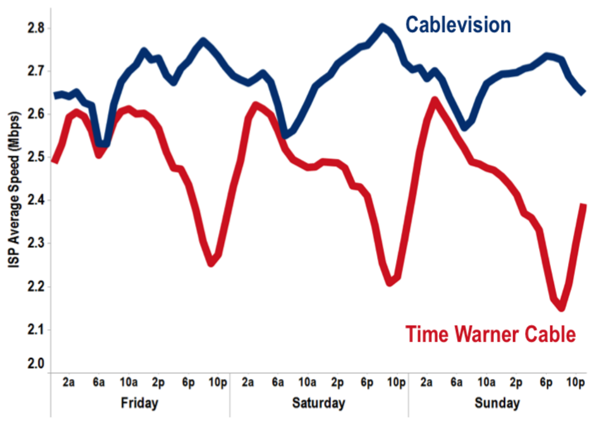
Not all ISPs want to adopt the Netflix Open Connect caching solution, rooting in multiple arguments, such as:

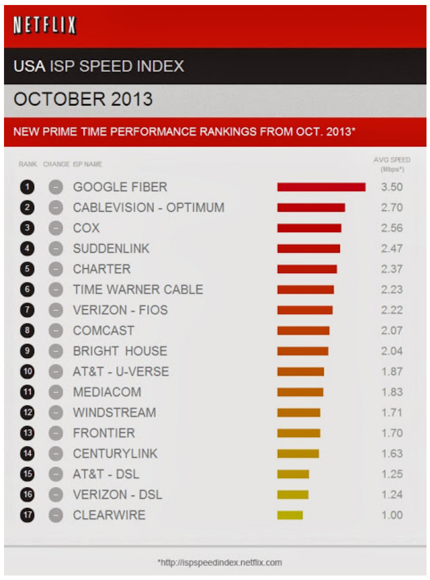
- Feel they are able too stream all available content qualities without deploying caches [Spangler, 2013]. This argument was from the time, when Netflix did not allow streaming Super HD, without an ISP adopting Open Connect.
- It should not be free to rent space for servers at the ISP's data centers [Thorkildsen, 2014a].
- OTT content providers should contribute to costs in developing an infrastructure capable of rising traffic demands, since they occupy a large amount of the available capacity [Thorkildsen, 2014b].

This has lead to deals being made privately between ISPs and Netflix, where Netflix pays the ISPs in order to place servers at their co-location facilities and obtain a more direct connection to the ISP's network [Rayburn, 2014]. Two of such ISPs are Comcast (USA) and Telenor (Norway). This was a much needed upgrade for the Netflix subscribers associated to these ISPs as the performance was decreasing each month, where Comcasts speeds were decreasing from Oct. 2013 to Jan. 2014. The deals sound a lot like deploying Open Connect caches, but with expenses associated with renting space at their co-location facilities. A possible reason for Netflix and the ISPs to seal the details of the deals are that Netflix still want to try and acquire free deployments at ISPs by only paying for costs associated with the equipment and both parties not wanting to reveal the details relating to pricing of such an agreement.

Pressuring ISPs to adopt Open Connect CDN

Netflix have with different methods tried to promote and pressure ISPs to adopt the Open Connect CDN. One of its initial steps were to host ispspeedindex.netflix.com. This site posts a monthly update on how well different ISPs are equiped to receive Netflix content in a measure of average stream speed during prime time. This allows an ISP subscriber to investigate which ISP will provide the best Netflix quality and make the choice of which ISP to subscribe to.




NETFLIX

USA ISP SPEED INDEX

OCTOBER 2013

NEW PRIME TIME PERFORMANCE RANKINGS FROM OCT. 2013*

RANK	CHANGE	ISP NAME	AVG SPEED (Mbps*)
1	–	GOOGLE FIBER	3.50
2	–	CABLEVISION - OPTIMUM	2.70
3	–	COX	2.56
4	–	SUDDENLINK	2.47
5	–	CHARTER	2.37
6	–	TIME WARNER CABLE	2.23
7	–	VERIZON - FIOS	2.22
8	–	COMCAST	2.07
9	–	BRIGHT HOUSE	2.04
10	–	AT&T - U-VERSE	1.87
11	–	MEDIACOM	1.83
12	–	WINDSTREAM	1.71
13	–	FRONTIER	1.70
14	–	CENTURYLINK	1.63
15	–	AT&T - DSL	1.25
16	–	VERIZON - DSL	1.24
17	–	CLEARWIRE	1.00

*http://ispspeedindex.netflix.com

Figure 3.14: Netflix Prime Time ISP Performance Metrics (Left figure) from Evers [2013] two ISPs showing the differences to using Open Connect CDN with caching (Cablevision) and not (Time Warner Cable). An ISP speed index (right figure) shows the avarage Netflix streaming speed for the different ISPs during prime time.

Figure 3.14 shows the content of a blog post, where the average Netflix streaming speed of two ISPs are compared during prime time and the ISP speed index of October 2013 is shown. In this case, Cablevision has adopted the Open Connect CDN, while Time Warner Compant has not. It is clearly seen that by having caches, Cablevision is able to handle the prime time traffic, while Time Warner Cable exhibits a decreasing average speed, until prime time is over.

Another method Netflix have used is to close access to their Super HD quality, if ISPs did not place their caches in their network [Netflix, 2013a]. Super HD was later opened to all its subscribers, but using adaptive methods to control the quality of the stream according to the available bandwidth.

Netflix cache planning

Planning of the caches is done in a collaboration between Netflix and the ISPs. There are different factors that determine the number of caches that are deployed and at what locations [Netflix, 2013b, p. 5]:

- Required content availability in region
- Amount of traffic handled by caches in region
- Chosen percentage of Netflix traffic one wants handled by a cache in the region

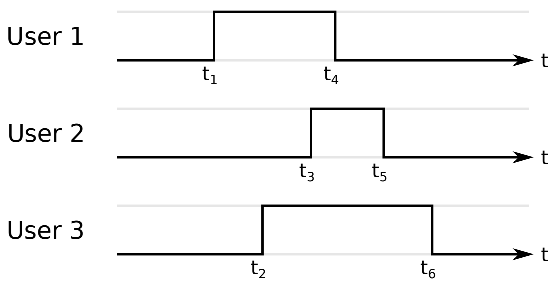
- Constraint to the traffic a cache can handle (7, 9 or 12 Gbps sustained throughput, dependent on appliance revision and configuration)
- Required amount of redundancy (placing caches in failure tiers)

These are all interesting points to analyse in order to make an informed choice as to where the cache can be placed in order to provide an offload of the ISP's network and avoid congestion.

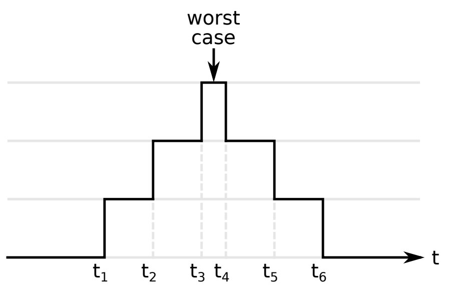
3.4 Assumptions and delimitations

This section will explain the different assumptions and delimitations that are performed in this thesis. The task is to find a good placement of caches on Bornholm to supply Netflix OTT content, specifically their Super HD video streams. The Bornholm scenario is presented in section 2.3, while the case study of Netflix is presented in section 3.3.

The cache plan focuses on being able to supply all demand from each CO on Bornholm. Though it is acknowledgeable that all inhabitants of Bornholm do not currently subscribe to Netflix, the cache plan will present a future-proof plan for a time where this is the case. Although the focus will be on planning for all of Bornholm, the algorithm can easily be altered to plan for another case by altering the NT population demands at each CO. Other cases could e.g. be predictions of the amount of Netflix subscribers at each CO. This would be a much more viable input to the algorithm, but since this information is not available, a good deployment of caches is found that supplies the complete demand of all COs.



(a) Users' service demand over the same link



(b) Aggregated demand of link over users

Figure 3.15: Worst case situation of a link's load

The cache plan satisfies a worst-case situation for the demand as seen in figure 3.15a and 3.15b, where all demands are active at the same time. The figures represent different users' usage of the same link and its aggregated usage. The worst case situation for the link load is given when all demands overlap. It can be seen that the on-period of the users' usages have different periods. If one had a case where these periods were small, the worst case situation would be very improbable, while if they were larger the case would have a higher probability of occurring. The plan focuses on satisfying the demand at this worst case situation by identifying a potential solution's maximum link load and finding a solution where it is minimised. If one wanted to guarantee a worst-case situation of a certain link load, a subset of NTs can be chosen by scaling the NT populations at each CO.

The following assumptions and delimitations are also performed:

- All demand is supplied by caches inside the network (not by gateways)
- Demand from COs are splittable
- All caches contain the same popular content
- The complete population of Bornholm act as Netflix subscribers.
- The plan does not focus on achieving a certain amount of redundancy if failure occurs
- Caches have a maximum capacity of 12 Gbps
- Caches are directed caches, meaning requests from NT's to caches are always successfully handled.
- Each NT produces a demand of 5800 Kbps corresponding to a Super HD stream.
- The ISP backhaul consists of a single one hop fiber links for each link.
- The layered structure of an ISP is not investigated.

The reason for gateways not being part of the search for a solution is that the percentage of OTT content requests leaving the network can be dynamic and indeterministic without a behaviour analysis that can identify worst-cases. On the other hand planning for a specific percentages of outgoing requests will only yield good solutions for this specific percentage and if it changes, the plan is no longer expected to provide as good an offload. A numerical evaluation is therefore performed after a solution is found. This numerical evaluation will allow for an investigation of how the found solution is affected when a percentage of the content is provided through gateways. This will allow the network planner to get insight into how the link loads can change in the network and allow one to identify links that can potentially become congested.

Demand is splittable, as this will allow the demand from a single CO to be supplied by more than one cache. This increases the complexity of the algorithm, but is necessary, so each CO can be given equal opportunity to access content from any cache in the network. It is also seen that the maximum demand is from CO 29 with a demand of 3005 NTs, which is approximately 16.6 Gbps. This will exceed the cache capacity of 12 Gbps. This can only be solved by allowing splittable demand.

The complete population of Bornholm acts as Netflix subscribers, because other information is not available regarding either the current of future estimates for Netflix subscribers. If GIS data was available, that could identify Netflix subscribers associated to each CO this could be used as the NT population. But, it may be more preferable to plan for the future case, specifically an estimate on the maximum number of expected Netflix subscribers. This will allow the plan to not be over-provisioned. When planning for these future cases, it is realistic to plan a progressive roll-out of these caches. This means that it may not be necessary to deploy all caches at once, but only supply the demand currently in the network. By progressively adding e.g. one cache at a time one could satisfy the progressive demand while being able to maintain optimality for future demands. If instead one was to plan caches one at a time, using the current deployment of caches, it would be a very cost inefficient solution in the future and may require additional caches to satisfy a certain maximum link load. Therefore by first planning for a future demand and progressively adding caches from this future cache deployment plan, one will ensure optimality for the future. A part of the problem then lies in deciding which caches should be chosen for each time additional demand is required. If the amount of caches is small it may be more efficient to perform an exhaustive search of the optimal sequence of caches to deploy, especially if it is one cache at a time. If one want to deploy multiple caches at a time, and find good deployments, then GAs may once again be applicable.

A delimitation is performed to focus on the case, where all caches contain the same content, which is also the case for current deployments of Netflix caches in Denmark. Redundancy is not the focus of this algorithm, as the case of all caches containing the same content will give high redundancy in itself. So there will be no specific tiering infrastructure. The caches are directed caches, meaning they only receive requests for content they have available and cannot perform any decisions as to indicate where content is available. This logic is performed by a central server when users requests a specific piece of content, the central server responds with the information on which cache the user must then request the content from.

Now that a delimitation has been performed in relation to the caches, the amount of caches required to supply the content can be calculated. The calculation is seen in equation 3.7 and is performed with information of the NT populations at each CO (N), bandwidth consumption from each NT (*Bandwidth*) and the cache capacity (*CacheCap*). The NT population at each CO are seen in figure 2.6 or 2.8, while the bandwidth and cache capacity are found through the case study of Netflix in section 3.3. The total NT population of the ring and double ring are identical, so the minimum amount of caches required to supply the demand is the same. The total NT population, $\sum_i^n(N)$, is 31998 NTs. This means, that when Netflix subscribers stream Super HD videos, each with a bandwidth of 5800 Kb/s, the total demand will become $31998 \cdot 5800\ Kb/s \approx 177\ Gb/s$. Which in turn means that if a cache can supply 12 Gb/s of demand, the minimum amount of caches is ≈ 14.75, meaning that 15 caches must be deployed to supply all of the demand.

$$MinCaches = \left\lceil \frac{\sum_i^n(N) \cdot Bandwidth}{CacheCap} \right\rceil = \lceil 14.7456 \rceil = 15 \tag{3.7}$$

A single fiber is used to construct a link between two caches for the scenarios, but if one wanted to incorporate the case where several fibers are used the complexity in the definition of the used adjacency matrix must be increase and redefined to allow this.

The results of this project will present the trade-off between the amount of caches deployed and how well the network is load balanced. An investigation will be performed for 15 to 20 deployed caches for both ring and double ring topologies. This should give a clear picture of how additional caches impact the link loads due to additional caches, and what differences there are between the two topologies.

A complete description of the scenario's delimitations are seen in table 3.5. It is also seen here that the $CacheCost$ is 1$ and $LinkCap$ is set to ∞. A caches cost is $1 as to easily map it to the number of caches deployed. This means that the designed algorithm will search for a good deployment of caches using information on the amount of caches instead of the summed cost of the caches. As these numbers are linear to each other, using costs will not affect the results. $LinkCap$ is set to ∞ because there is no reason to see solutions that exceed the link capacity as infeasible, since the whole objective of the algorithm is to find a deployment of caches where the maximum link load is as small as possible. A $LinkCap$ can instead be used to evaluate the solution by identifying where one may be required to place additional fibers to handle the load.

Description	Value
$Bandwidth$	5800 Kb/s
$CacheCost$	1
$MinCaches$	15
$MinCachesCost$	15
$LinkCap$	∞
$CacheCap$	12 Gb/s

Table 3.5: Delimitations of scenario

Chapter 4

Method

This chapter describes the method for applying a genetic algorithm to the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem. The chapter also includes a method for numerically evaluating a solution where OTT content is provided through both caches and gateways. Thereafter the genetic algorithm is tested using known global optimum solutions. Lastly, advantages and limitations are recognised. Figure 4.1 shows the flow of the method in a flow chart, where each process/decision is grouped into either input, output or Genetic Algorithm (GA).

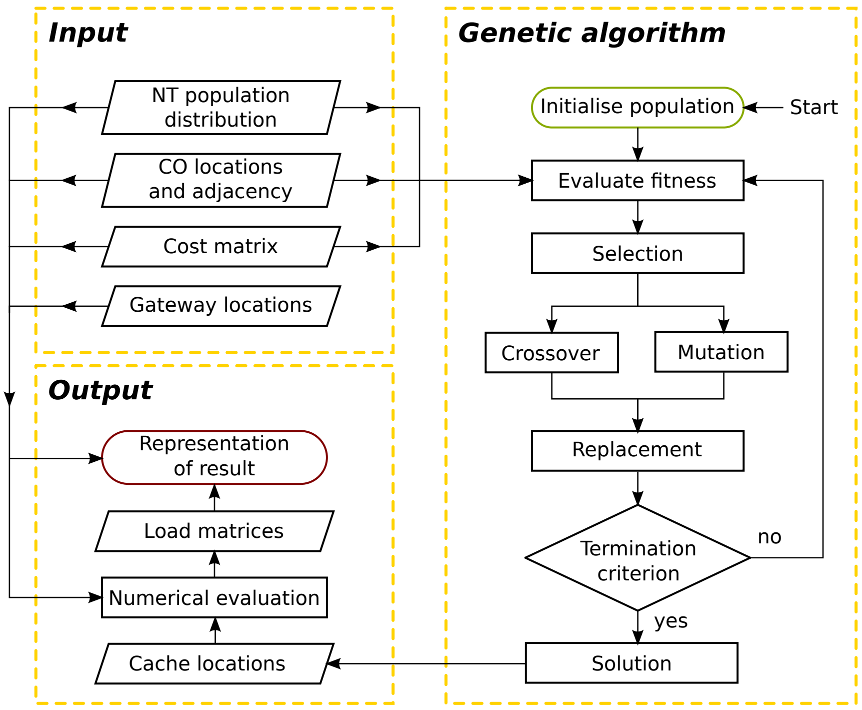


Figure 4.1: Flow chart of the method

4.1 Preparation of input data

For the genetic algorithm to evaluate the fitness, it requires an objective function. This objective function uses GIS input data from which a routing algorithm can estimate the link loads across the network. The GIS data consists of NT populations (N), CO locations (X, Y), CO adjacency (A), cost matrix (C) and gateway locations. The following sections will describe how the adjacency matrix, cost matrix and gateway locations are found and represented. The NT populations and CO locations have already been described in the scenario (see section 2.3) and are therefore not described in this section. While the adjacency is shown in the scenario, the method for constructing the adjacency matrix it is not. It is therefore described in the following.

4.1.1 Adjacency of central offices

The adjacency of COs is used to calculate the routes from each NT to caches and for the visual representation of the CO inter-connection. As was seen in the generic definition, the elements of the adjacency matrix can be written as:

$$a_{ij} = \begin{cases} 1, \text{if connected} \\ 0, \text{otherwise} \end{cases} \quad \text{, where } i, j \in 1..n \tag{4.1}$$

As described in the scenarios (see section 2.3), the ring and double ring topology are used. These two topologies have a certain adjacency matrix that clearly identify the type of topology. These can be constructed using shifted diagonal matrices as seen in equation 4.2. These diagonal matrices are defined by their size and a positive/negative shift for defining where the diagonal occurs in the matrix. A positive shift, shifts it towards the upper right corner of the matrix, while a negative shift, shifts it towards the lower left corner of the matrix.

$$diag(len, shift) : \text{Shifted diagonal matrix of size } len \times len \tag{4.2}$$

$$shift : \text{Value for shift of diagonal} \tag{4.3}$$

The ring topology is constructed using equations 4.4, 4.5and 4.6 in consequetive order. Table 4.1 shows an example of the ring adjacency matrix for $n = 10$.

$$A_R = diag(n, 1) + diag(n, -1) \tag{4.4}$$

$$A_R(n, 1) = 1 \tag{4.5}$$

$$A_R(1, n) = 1 \tag{4.6}$$

$$n : \text{Amount of COs} \tag{4.7}$$

		From									
		1	2	3	4	5	6	7	8	9	10
To	1	0	**1**	0	0	0	0	0	0	0	**1**
	2	**1**	0	**1**	0	0	0	0	0	0	0
	3	0	**1**	0	**1**	0	0	0	0	0	0
	4	0	0	**1**	0	**1**	0	0	0	0	0
	5	0	0	0	**1**	0	**1**	0	0	0	0
	6	0	0	0	0	**1**	0	**1**	0	0	0
	7	0	0	0	0	0	**1**	0	**1**	0	0
	8	0	0	0	0	0	0	**1**	0	**1**	0
	9	0	0	0	0	0	0	0	**1**	0	**1**
	10	**1**	0	0	0	0	0	0	0	**1**	0

Table 4.1: Adjacency matrix of ring topology, $n = 10$.

The adjacency of the double ring, A_{DR}, is constructed by constructing two smaller adjacency matrices, $A_{DR,A}$ and $A_{DR,B}$. $A_{DR,A}$ is constructed by performing three operations consecutively. First the diagonals are created using equation 4.8. Then the upper-right and lower-left corners of $A_{DR,A}$ are set to one, as seen in equation 4.9 and 4.10. $A_{DR,B}$ is constructed in equation 4.11. A_{DR} is then constructed by combining $A_{DR,A}$ and $A_{DR,B}$ as seen in equation 4.12. Table 4.2 shows an example of the double ring adjacency matrix for $n = 10$.

$$A_{DR,A} = diag(n/2-1,1) + diag(n/2-1,-1) \tag{4.8}$$

$$A_{DR,A}(n/2,1) = 1 \tag{4.9}$$

$$A_{DR,A}(1,n/2) = 1 \tag{4.10}$$

$$A_{DR,B} = diag(n/2-1,1) \tag{4.11}$$

$$A_{DR} = \begin{bmatrix} A_{DR,A} & A_{DR,B} \\ A_{DR,B} & A_{DR,A} \end{bmatrix} \tag{4.12}$$

		From									
		1	2	3	4	5	6	7	8	9	10
To	1	0	**1**	0	0	**1**	**1**	0	0	0	0
	2	**1**	0	**1**	0	0	0	**1**	0	0	0
	3	0	**1**	0	**1**	0	0	0	**1**	0	0
	4	0	0	**1**	0	**1**	0	0	0	**1**	0
	5	**1**	0	0	**1**	0	0	0	0	0	**1**
	6	**1**	0	0	0	0	0	**1**	0	0	**1**
	7	0	**1**	0	0	0	**1**	0	**1**	0	0
	8	0	0	**1**	0	0	0	**1**	0	**1**	0
	9	0	0	0	**1**	0	0	0	**1**	0	**1**
	10	0	0	0	0	**1**	**1**	0	0	**1**	0

Table 4.2: Adjacency matrix of double ring topology, $n = 10$.

4.1.2 Cost matrix

The cost matrix, C, describes the shortest path distances in a route between a CO and a cache using either hop count or euclidean distance between the COs (using their geographical locations). The cost matrix is used by the shortest path first routing algorithm of the objective function (see section 4.2.2) to allow it to distinguish between the costs of utilising different routes in the network. The costs were previously defined in the generic definition as:

$$c_{ij} = \begin{cases} \mathbb{R}^+, \text{if } s_j = 1 \\ 0, \text{otherwise} \end{cases} \quad \text{, where } i, j \in 1..n \tag{4.13}$$

If the cost matrix is constructed using the euclidean distance between each CO and the adjacency matrix. The operations to calculate the euclidean distances and costs are respectively seen in equations 4.14 and 4.15. The cost matrix, C, is an element-wise multiplication of the adjacency matrix, A, and the distance matrix, D.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.14}$$

$$C = A \circ D \tag{4.15}$$

4.1.3 Gateway locations

Gateways are required for numerical evaluation. Therefore the gateway locations are identified by investigating submarine cable maps provided by TeleGeography [2014]. This thesis focuses on the scenario of Bornholm, where there exists two of such gateway locations. Figure 4.2 shows these gateways depicted by their geographical location.

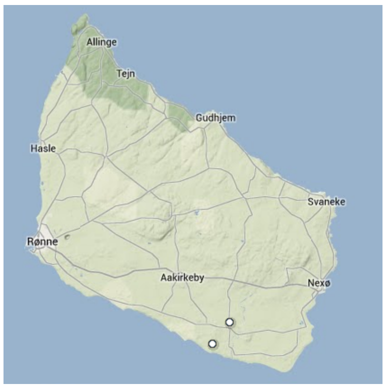
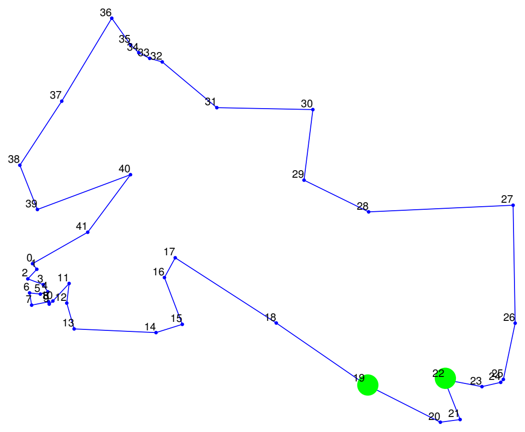


Figure 4.2: The two gateways located in Bornholm (white dots) [TeleGeography, 2014]

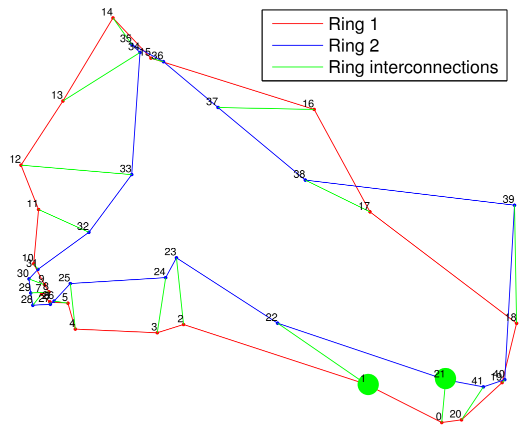
The two COs with shortest euclidean distance are chosen as gateways. The CO IDs are dependent on the topology, therefore the gateway IDs will be different dependent on the used topology. The gateway IDs are seen in table 4.3, while their location in the topologies is seen geographically in figure 4.3 and logically in figure 4.4.

Topology	Gateway IDs
Ring	19, 22
Double ring	1, 21

Table 4.3: Gateway IDs for the two topologies

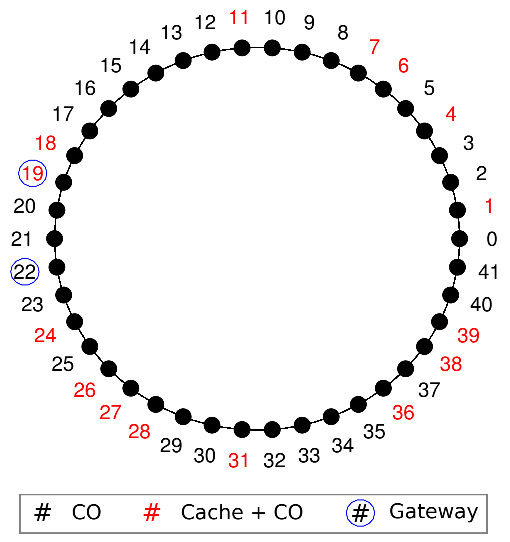


(a) Ring

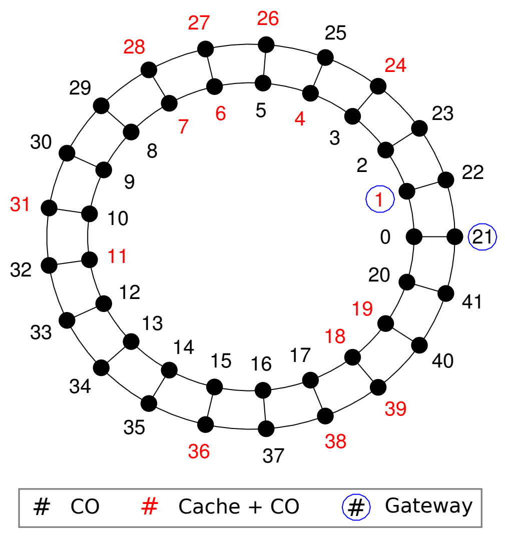


(b) Double ring

Figure 4.3: Geographical plot of topologies with their 2 specified gateways (green dots)



(a) Ring



(b) Double ring

Figure 4.4: Logical plot of topologies with 2 specified gateways and arbitrarily placed caches

4.2 Genetic algorithm

This thesis works with Matlab's global optimisation toolbox [Matlab, 2014b]. It provides a modular framework for multiple optimisation algorithms including Genetic Algorithms (GAs), making it easy to add to/edit each step of the GA (see figure 4.1). It also provides a handful of generic reproduction operators and a plotting environment to follow and analyse the evolution while it is executing.

The following sections will describe how the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem is solved using a GA when planning OTT caches in an ISP's network. It should be noted that the thesis seeks to solve the LLB-CFL problem for a real-world scenario, meaning that all parameters and settings for the GA are found for the specific scenario of Bornholm, which has 42 COs and a specific population of NTs at each of them (see section 2.3).

4.2.1 Population

The concept of a population in a GAs is introduced in section 3.2.3. A population consists of individuals that each describe where a set of caches are deployed. Each individual, S, is represented using a bit string. The bit location indicates the location in the topology (see figure 2.6 and 2.8), while the bit value indicates if a cache is deployed at the location or not.

In the provided scenario, the amount of COs, which is 42, defines the length of the bit string, or in other words the amount of potential cache locations. An example individual with 15 caches is shown in table 4.4. The relation between this example bit string and the ring topology is seen in figure 4.5.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Value	0	**1**	0	0	**1**	0	**1**	**1**	0	0	0	**1**	0	0
Bit	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Value	0	0	0	0	**1**	**1**	0	0	0	0	**1**	0	**1**	**1**
Bit	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Value	**1**	0	0	**1**	0	0	0	0	**1**	0	**1**	**1**	0	0

Table 4.4: Individual with 15 caches deployed at {1, 4, 6, 7, 11, 18, 19, 24, 26, 27, 28, 31, 36, 38, 39}

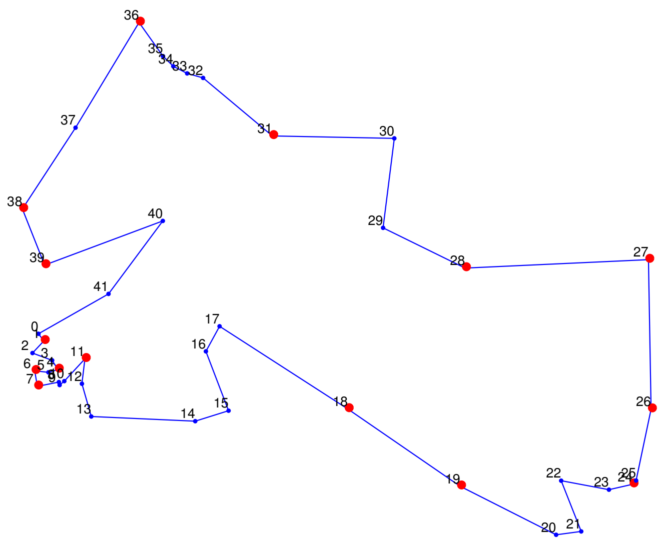


Figure 4.5: Plot of ring topology with caches deployed (red dots) according to bit string in table 4.4

Initial population

Due to the constraints of the *RequiredCost*, the amount of caches is fixed. This allows the GA to find an good solution for a given amount of caches, one at a time. Therefore the individuals in the initial population must be generated to contain the same amount of caches. The initial population must have a large diversity, allowing it to encompass a large part of the solution space. Therefore random initialisation is used. The function is for initialising the population is described in algorithm 1.

Algorithm 1 Initialise population

```
1:  n: Amount of COs
2:  p: population size
3:  c: Amount of caches
4:  function InitialPop($n, p, c$)
5:      $Population \leftarrow zeros(p, n)$        ▷ Construct $p$ empty individuals of length $n$
6:      for $i = 1 : p$ do                        ▷ Construct $p$ individuals
7:          $perm \leftarrow$ random permutation of the integers from 1 to $n$ inclusive
8:          $locations \leftarrow perm(1 : c)$     ▷ Choose $c$ out of $n$ locations
9:          $Population(i, locations) \leftarrow 1$   ▷ Deploy caches at these locations
10:     end for
11: end function
```

Population size

The population size indicates how many individuals are in a generation. There exists a trade-off between the quality of solution and computational cost. An investigation of the population size is performed by fixing all other GA parameters (as seen in table 4.5) except the population size. Five runs are performed for each population size using different random seeds. Figure 4.6 shows the trade-off between mean running time and mean fitness of different population sizes. It is seen that the best solutions are generated using a population size of 400 as it provides the best trade-off. It has a reasonable computation time, while improving on the mean fitness. The best fitness is equal for all population sizes, but a smaller mean fitness increases the probability of the planner achieving good results with few runs. Increasing the population size further will not provide an improvement to fitness, but will increase the computational cost.

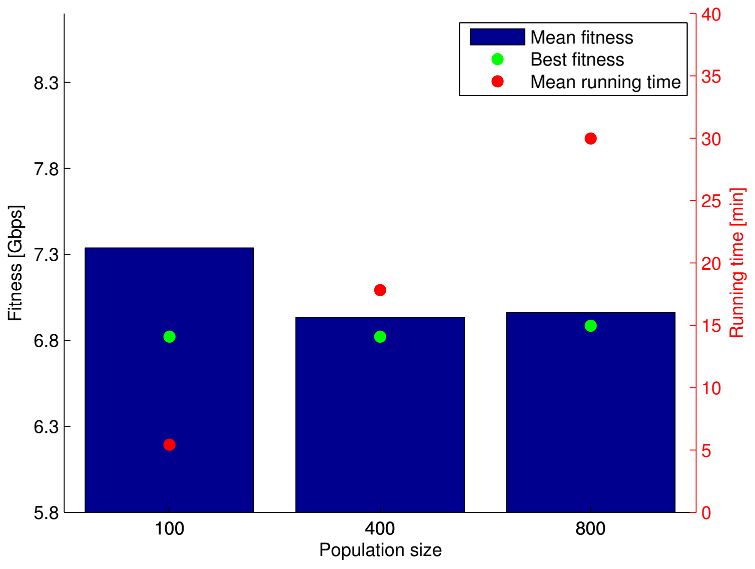


Figure 4.6: Mean fitness, best fitness and running time for different population sizes

4.2.2 Objective and fitness function

In GAs an objective and fitness function have different jobs. The objective function uses the deployment of caches to apply a routing algorithm and supplies a load matrix, L. This load matrix contains information on the cache and link loads as is described in the generic definition in section 2.1.4. The fitness function evaluates this load matrix by identifying the maximum link load, l_{max}. This will indicate how well the network is link load balanced or in other words, how robust the network is against unpredictable traffic growths.

The implementation of the GA allows interchanging the objective function to fit the planner's needs. The requirements to the implementation of it is that it provides a load matrix and ensures that the capacity of the caches are not exceeded. If the routing algorithm does not ensure the capacity of the caches are not exceeded the fitness must account for it by introducing a penalty. Algorithm 2 shows the pseudo-code for the general process of the objective function. Appendix B contains the pseudo-code for each of the functions utilised in the objective function.

Algorithm 2 Objective function

```
S: Individual / cache deployments
n: Amount of COs
N: NT population vector (demand of each CO)
A: Adjacency matrix
X,Y: xy-coordinate vectors
PopFrac: Fraction of NT population vector to add at each loop
Bandwidth: Bandwidth of OTT content traffic from each NT
CacheCap: Maximum sustainable throughput from a cache

function ObjectiveFunction($S, n, N, A, X, Y, PopFraction, Bandwidth, CacheCap$)
    - Find all paths and their OD distance from all COs to each deployed cache
    $P, C = PathsAndDist(A, X, Y)$
    - Associate NTs to caches using round robin and shortest path first
    $Assoc, CacheLoads = Associate(n, N, PopFrac, Bandwidth, CacheCap)$
    - Find load matrix describing cache and link loads
    $L = CalcLoadMatrix(n, P, Assoc, CacheLoads)$
    return $L$
end function
```

The idea behind this objective function is to mimic the functionality of the Open Shortest Path First (OSPF) routing protocol, which is a protocol used for routing traffic internally in a ISP's network. The routing is performed by first calculating each CO's shortest path route each cache. This can be performed using an adjacency matrix and xy-coordinates of the COs. In this thesis hop-count is used, meaning that the distances between COs are set to 1. A round robin procedure will stepwise route a fraction of each CO's NT population (*PopFrac*) to the shortest path cache. The routing algorithm will ensure that cache capacities are not exceeded. This is done by taking the next shortest path available, when the capacity of the cache reached through the shortest path capacity is full. *PopFrac* is chosen to be 0.03 (3%) as it shows the smallest average computation time. Because of the non-uniformity of the NT populations, there will almost always be a case where the 3% cannot be satisfied by a cache. To ensure that the demand can be satisfied anyway, a demand of 1 is supplied at this iteration. When the cache capacity is filled up, the routing algorithm will then try to supply 3% of a CO's demand from the next shortest path cache. A choice of a *PopFrac* of 0.03 ensures that for this scenario, the cases where the remaining cache load is filled with 1 demand at a time, is minimised and therefore less computation time is used.

When all demand has been associated to caches, the load matrix can be calculated and the fitness can be evaluated. The fitness is evaluated by first ensuring that the load matrix is an undirected representation of the loads (summing upper and lower triangular) and then identifying the maximum link load.

4.2.3 Selection

Selection is the process of choosing which individuals from the current population that must reproduce. It is described in detail in section 3.2. Rank fitness scaling is performed on the fitnesses in order to allow low fitness individuals a reasonable probability of selection. 20 elite individuals has shown to give better results compared to using no elite individuals. The reason behind this could be that a certain set of good fitness individuals have good gene material for later generations.

A method for choosing the selection function is performed by analysing the mean and best fitness. Five runs are performed for each selection function with different random seeds using the parameters in table 4.5. The mean fitness indicates how good solutions the selection function provides over multiple runs with different random seeds. The best fitness shows the best found solution from each selection method. Figure 4.7 shows a comparison between tournament, stochastic uniform sampling and roulette wheel. They all indicate an ability to provide equally optimal solutions, while roulette wheel is better at providing good solutions in general. This means that by using roulette wheel as the selection method for this problem, there is a higher probability of generating a better fitness solution.

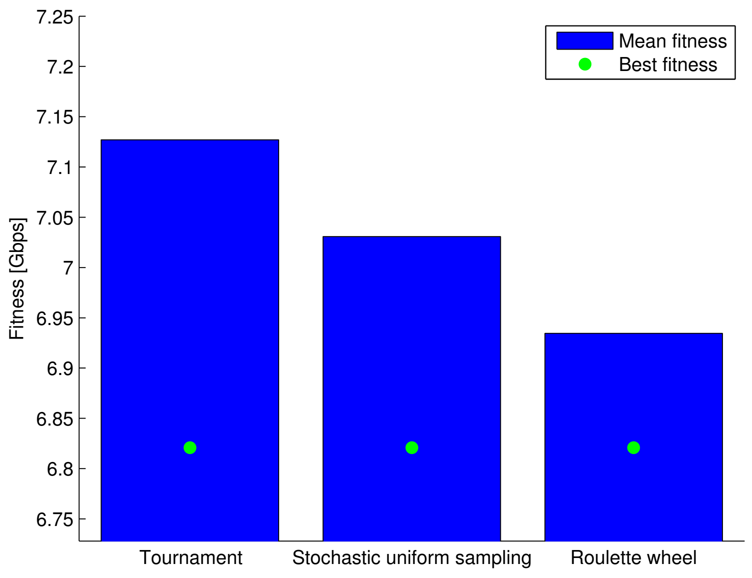


Figure 4.7: Comparison of selection methods

4.2.4 Crossover and mutation

Together with selection, evolution is performed using crossover and mutation. Crossover operations combines good fitness individuals to produce better individuals, while mutation provides diversity to avoid termination at local optima solutions.

Crossover function

Due to constraints to the amount of caches through *RequiredCost*, generic crossover functions such as one point, two point, N point and uniform crossover can not be used. These crossover functions do not guarantee that the amount of caches are unaltered. Exchanging is therefore a viable alternative, as it guarantees that the amount of caches is the same after the operation is performed. The exchange crossover function is described in the analysis of GAs (see section 3.2).

Crossover fraction

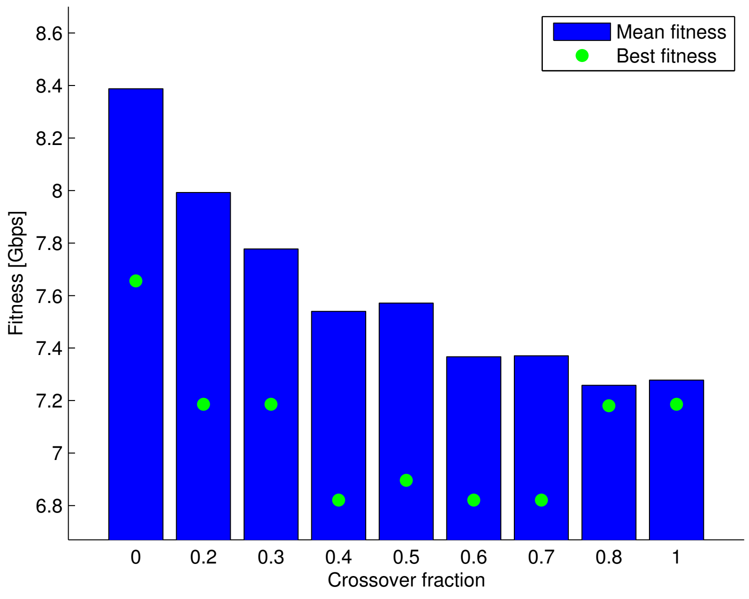


Figure 4.8: Comparison of crossover fractions

A crossover fraction indicates how much of the population is used for crossover, while the remaining fraction is used for mutation. Figure 4.8 shows an analysis of different crossover fractions, where the mean fitness indicates how good solutions are provided. The best fitness indicates how good the GA is at finding good fitness solutions with different crossover fractions. By looking at the crossover fraction from 0 to 1, one can see the extremes of the two cases. The analysis is performed over 5 runs with different random seeds and for 20 generations. The GA evolution is only 20 generations long, because this method investigates the efficiency of choosing a specific crossover fraction. With a crossover fraction of 0 there is no crossovers being performed, only mutations, therefore a too high diversity is achieved and there is no progression. On the other hand a crossover fraction of 1 provides better solutions due to progression through crossover, but exhibits premature convergence due to the lack of mutation. A crossover fraction of 0.6 is chosen as it both achieves the *best* best fitness and a good mean fitness, meaning that it provides the best efficiency.

Mutation functions

Alike crossover, the generic mutations functions such as flipping does not guarantee to uphold the *RequiredCost* constraints, therefore the viable alternatives are interchanging and heuristic hypermutation. Normal interchange often got stuck in local optimum solutions, when using different random seeds, and hypermutation increased the computation time excessively (running times over 24 hours) without providing a good efficiency. Both mutation functions are detailed in the analysis of GAs in section 3.2.

It was observed that multiple solutions were very close (2-3 CO locations off for planning of 15 caches) to the best found solutions, but were unable to reach it. Using a uniform selection of the amount of caches to interchange created too much diversity, therefore a probability based multiple interchange mutation function is designed to provide increased diversity to the population allowing more than one change to occur and provide better avoidance of termination at local optima solutions. The multiple interchange mutation function has two parameters; maximum interchanges, *MaxInterchanges*, and a scaling factor, α. Together they define a discrete function that indicates the probability of interchanging up to *MaxInterchanges* cache locations. The discrete function follows the following two rules:

- Probability of interchanging a certain amount of caches, *ChangeAmount*, is proportional to $1/ChangeAmount^{\alpha}$
- Probabilities are scaled, so their sum is 1

The pseudo-code for performing the probability based multiple interchange mutation is seen in algorithm 3. Different values of *MaxInterchanges* and α and the corresponding probabilities of interchanging different amounts of caches is depicted in figure 4.9. Figure 4.10 shows a comparison in fitness for the normal interchange (*MaxInterchanges* = 1) a handful of parameters for the multiple interchange function. There is a significant difference by applying multiple interchanges to the mutation, increasing both the mean and best fitness. It is also seen, when comparing the parameters, that there is little to no difference in choosing *MaxInterchanges* ≤ 3 and any value of the scaling factor α, while *MaxInterchanges* > 3 introduces too much diversity. The analysis of the mutation function was performed with 3 different random seeds for each set of parameters. The low number of runs were used to save running time of the evaluation, as it took over 10 hours to complete. It is still able to show the general tendency in choosing the different types of parameters.

Algorithm 3 Mutation by a probability based multiple interchanging

```
1: S: a bit string individual where a one indicates a cache deployment
2: function ProbMultipleInterchange(S)
3:     for $ChangeAmount \leftarrow 1 : MaxInterchanges$ do
4:         $Prob(i) \leftarrow 1/(ChangeAmount^{\alpha})$
5:     end for
6:     $Prob \leftarrow Prob/sum(Prob)$                ▷ Normalise sum of $Prob$ to 1
7:     $Prob \leftarrow cumsum(Prob)$                  ▷ Compute cummulative sum
8:     $ChangeAmount \leftarrow$ Generate uniform random variable and choose amount of in-
   terchanges according to the slot of $Prob$ it falls into
9:     Interchange $ChangeAmount$ amount of caches in individual $S$
10: end function
```

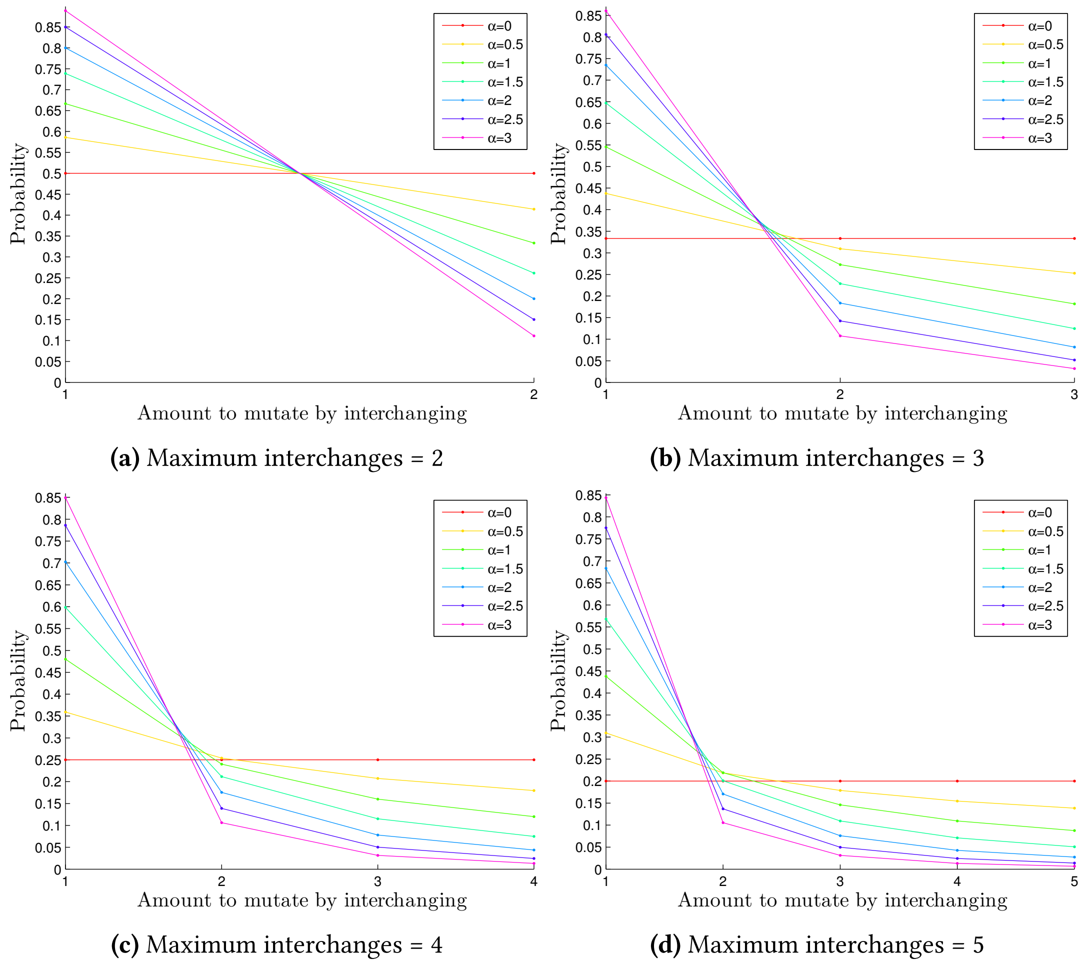


Figure 4.9: Probability based multiple interchange with different maximum interchanges

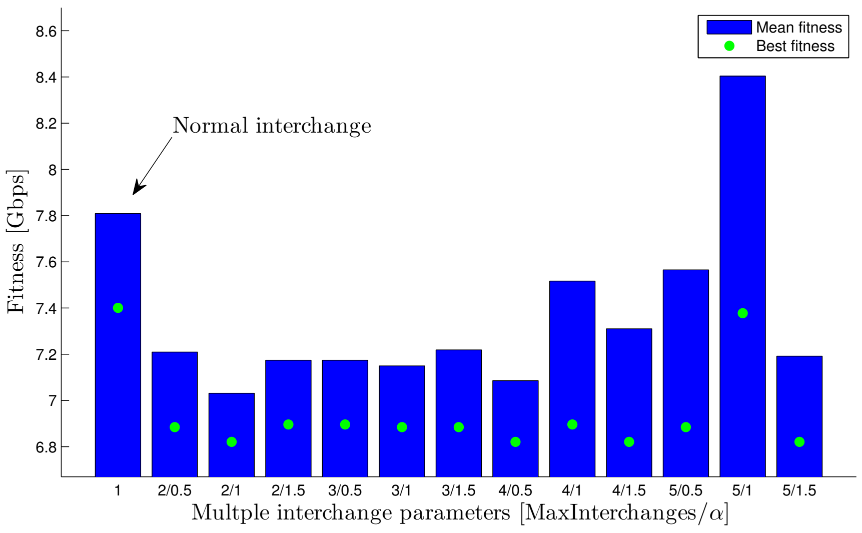


Figure 4.10: Different parameters for the probability based multiple interchange mutation function

4.2.5 Visualising the evolution of the Genetic Algorithm

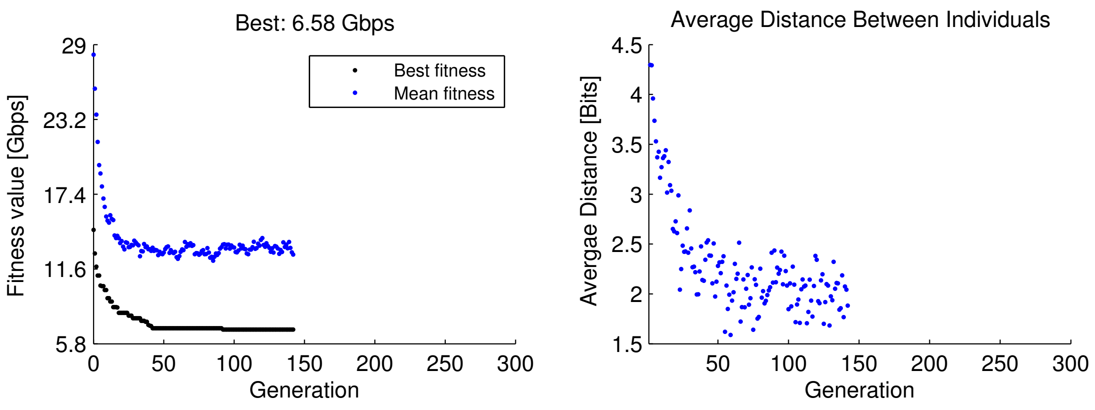


Figure 4.11: Example of evolution of the GA for 15 caches

The evolution of the GA is seen in figure 4.11, which depicts two of Matlab's plotting functions from their optimisation framework. On the left hand side of the figure, the mean and best fitness are depicted. This shows the gradual evolution of the algorithm, when it finds new and fitter solutions. The mean fitness shows that the diversity is maintained. If it was not maintained, the mean fitness would converge towards the best fitness. On the left hand side of figure 4.11, the average distance in bits between individuals is seen. This is calculated using euclidean distance and is a measure of similarity. As to not impact the efficiency of the GA significantly, 20 pairs are chosen at random from the population to calculate the euclidean distance, from where an average of these values is calculated. The convergence of the average distance between individuals indicates that although the solutions are diverse in fitness, the bit string representations are becoming more and more similar. The diversity is also seen in the average distance between individuals by the spread of the values.

4.2.6 Termination criterion

The termination criterion is evaluated through the amount of stall generations. The amount of stall generations can be decided by running the GA for a certain number of generations and identifying the stall point. The maximum amount of stalls experienced before reaching this point will define the amount of stall generations allowed. Increasing this value may give bigger confidence in the solution, but requires potentially unnecessary computation costs. Figure 4.12 shows how the stall generations can be identified. In this case we can see the maximum amount of stalls experienced before the final stall is 45 generations. The amount of stall generations is therefore chosen to be 50 generations, to still allow the final stall to occur. The amount of stalls seen in figure 4.12 is usually at or under the observed amount, and to avoid unnecessary computation costs it is set as close to the stall point as possible.

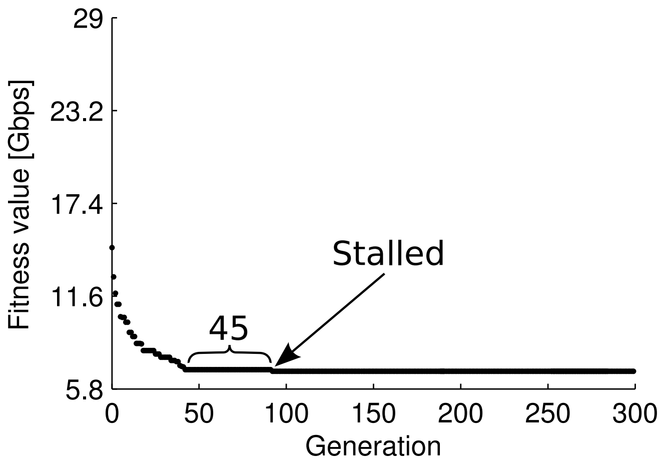


Figure 4.12: Finding the amount of stall generations

Instead of using a stall generation for termination a high maximum generations can be used. This will result in a very computationally expensive GA, as it will find solutions close to the best found solutions but only after 1000 generations (5.1 hours). A better solution is achieved using less amount of time, by running the GA multiple times with different random seeds. The algorithm will have achieved a smaller running time and produces better results with greater confidence. It produces better results, because the long runs still result in local optimum solutions worse than the best found using multiple runs with different random seeds. A single run takes approximately 17 minutes. The algorithm has shown to be able to find a good solution that are close to each other within 3-5 runs, taking around an hour to compute. Although most solutions are close to the best found solution, using this method, there are sometimes outliers that have too high a fitness. This indicates that mutation does not introduce a high enough diversity for some specific local optima, but does introduce sufficient diversity for many local optima. Increasing amount of mutation only for higher diversity will not positively affect the GA. Therefore if a better solution was to be found in one run, it could require increasing diversity after a certain amount of stall generations, for an amount of generations. This has not been investigated, and is seen as future work for improving on the method and guaranteeing the best found solution potentially being provided in one run.

4.2.7 Parameters of the Genetic Algorithm

From the performed investigations of different parameters of the processes in the GA, the parameters in table 4.5 are chosen for generating results.

Parameter	Value
Population size	400
Population fraction	0.03 (3%)
Maximum generations	300
Stall generations	50
Elite count	20
Fitness scaling	Rank
Selection function	Roulette wheel
Crossover function	Exchange
Crossover fraction	0.6
Mutation function	Prob. based multiple interchanging
Mutation parameters	$MaxInterchanges = 3$, $\alpha = 1$

Table 4.5: Paramters of the GA

4.3 Cost-fitness trade-off

A trade-off between the cost of a solution and the maximum link load (fitness) is investigated by performing a run of the GA on different amounts of caches. The amount of caches (*RequiredCost*), will take the values from 15 (minimum) to 42 in order to show an expected progressive decrease in maximum link load, when the amount of caches are increased, to a point where it will no longer provide a significant advantage (diminishing returns). This will allow a planner to both see the impact on the fitness by increasing the amount of caches and make a choice on how much he/she wants to spend on the solution.

4.4 Numerical evaluation of utilising both caches and gateways

After an optimal solution is found using the GA with the proposed parameters, a numerical evaluation of gateway traffic is performed. The numerical evaluation seeks to include the concepts of the gateway and content availability. The optimal solution is found for the worst-case scenario (see figure 3.15), where everyone 100% of the population in the network is requesting content that is available in the caches. But what about the scenario, where content is unavailable in the caches for a certain fraction of the population. This numerical evaluations seeks to provide information on how the links loads change for different content availability. Content availability is defined as the fraction of the each NT population, whose content is serviced by caches. So if the content availability is 80%, then 80% of the populations content is serviced by caches and 20% are serviced through gateways.

The numerical evaluation first constructs two vectors from the NT population vector, N. One is the vector, N_C, which is a scaled amount of N corresponding to the content availability fraction. It describes the population of NTs at each CO, whose demand is handled by caches in the ISP's network. The second vector, N_G, is a scaled according to 1 minus the content availability fraction. Using the obtained optimal solution and the two new population vectors, two individual load matrices are calculated. One for loads due to cache traffic and one due to gateway traffic. These two link load matrices are then summed to provide what is known as the total link load matrix. The total link load matrix provides the planner with knowledge on where potential congestions are more likely to occur.

4.5 Test of genetic algorithm

This sections provides simple examples of known global optima solutions for ring topologies, and applies the GA to the same problem to see if it finds the expected global optimum solutions. This will indicate how efficient the algorithm is at finding the known optimum solutions. The following simple case is presented:

- Each CO has a equal demand (NT population) of 2 that can be split up into two flows of demand 1
- For a graph with c caches, and m steps (see figure 4.13) between them, the amount of COs is $n = c \cdot m$
- Each cache has a capacity of $CacheCap = (demand \cdot n)/c$
- The amount of k-combinations are $n!/(c!(n - c!))$

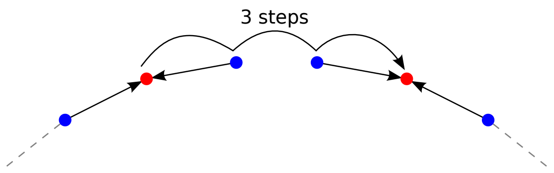
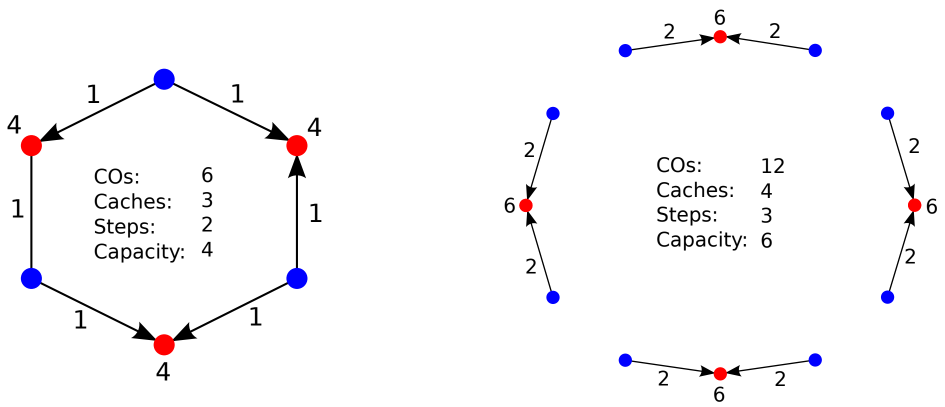


Figure 4.13: Example of steps in a ring with caches (red) and COs (blue)



(a) Ring with maximum link load of 1 **(b)** Ring with maximum link load of 2

Figure 4.14: Multiple interchange with different maximum interchanges.
Caches are red dots and COs are blue dots.

Figure 4.14 shows two of such graphs and table 4.6 shows the complete details for the graphs including the results of the GA. The genetic algorithm is provided with the adjacency matrix and the amount of caches. It must then find the optimal placement corresponding to a solution with caches spaced with equal distance steps between each other. It is seen that even with the case of 42 COs and 7 caches that it is possible to find the global optimal solution within few generations. This is an amount corresponding to 0.02% of the function evaluations when comparing with the exhaustive algorithm (k-combinations). This shows that the algorithm is capable of solving the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem within a very large solution space.

	Variable	Description	Values				
Given	c	Amount of caches	3	4	5	6	7
	m	Steps between caches	2	3	4	5	6
	$Demand$	Demand at a COs	2	2	2	2	2
Specifics	n	Amount of COs	6	12	20	30	42
	$CacheCap$	Cache Capacity	4	6	8	10	12
	$Poss$	k-combinations	20	495	15504	593775	26978328
Results	g	Generations	1	1	5	30	25
	l_{max}	Max. link load	1	2	3	4	5
	$FCount$	Function count	800	800	2400	10400	6000

Table 4.6: Test of genetic algorithm with known global optima solutions

4.6 Advantages and limitations

This section seeks to briefly emphasise the advantages and disadvantages/limitations in using the described method to plan caches in an ISP's network.

4.6.1 Advantages

The advantages are presented in the following.

Short running time to find an good solution.

In around an hour, a good solution that minimises the maximum link load can be found for a specific amount of caches. The run time increases linearly if one wants to compute the trade-off between cost and fitness.

Applicable for other related problems

The method can be applied to other use cases where one wants to balance link loads in a network and become better at avoiding congestion.

Modularity

The implementation and structure of the GA is highly modular, meaning that the routing algorithm, fitness evaluation, reproduction operators and many other processes can be altered or replaced completely without affecting other processes. Figure 4.1 shows the flow chart of the GA and which processes are replaceable.

4.6.2 Disadvantages / limitations

The disadvantages/limitations are presented in the following.

Global optimal solutions

This method and GAs in general can not guarantee that a found solution is globally optimal, but the optimisation algorithm instead finds good fitness solutions with a link load balanced network.

Parameter tweaking

The method requires a large amount of parameter tweaking to provide a good efficiency of the GA. Therefore a large amount of time has been spent in identifying which parameters (genetic operators) are better and how they should be configured.

Dependency on the routing algorithm

The provided method and its parameters may not work as well with other routing algorithms. Therefore an analysis of the methods may need to be performed to investigate the differences in efficiency, if there are any.

Chapter 5

Results

This chapter will contain two sets of results for both ring and double ring topologies:

1. A presentation of the link loads before and after deploying caches.
2. A cost-fitness trade-off analysis.
3. A numerical evaluation of the link loads, when a percentage of the content is handled through gateways.

A presentation of the best individuals found for the case of 15 caches for a ring and double ring topology shows that the GA is capable of finding good solutions. A comparison of the solution found using the described method (see section 4) with the case where all OTT traffic is handled through gateways, shows the before and after picture of caching OTT content. A cost-fitness trade-off analysis shows that when the amount of caches is unknown, a cost-fitness trade-off analysis can be performed to find a suitable amount of caches to deploy that will satisfy a required maximum link load. A numerical evaluation of the link loads, when a percentage of the content is handled through gateways will allow the cache planner to investigate potential congested links under different content availabilities. Content availability is defined as the fraction of requests handled by caches in the network. The resulting total link loads from combining the loads of traffic handled by caches and gateways will allow the planner to identify where in the network potential congestions are most likely to occur.

5.1 Before and after deploying caches

The best individuals are found for the minimum required amount of caches (15 caches, see section 3.4) of the two topologies are seen in figures 5.1 and 5.2. These figures show a comparison of before and after deploying caches. One where all OTT content is serviced through gateways and another where all OTT content is serviced by caches. The two gateway locations are placed as described in section 4.1.3. Figures D.1 and D.2 in appendix D show the corresponding cache loads for the two topologies.

It can immediately be identified that the link loads on both the ring and double ring topology show a vast improvement in the maximum link load. The ring topology shows a maximum link load of 6.82 $Gbps$, while the double ring shows a maximum link load of 5.13 $Gbps$. This shows an improvement corresponding to achieving a maximum link load 7.1% of what it was with purely gateway traffic for the ring topology and 12.3% for the double ring topology. A bigger improvement is seen for the ring topology, as it is more affected by the gradual increase in link loads towards the gateways, while the double ring has an additional *lane* to propagate traffic across. This additional connectivity of the double ring, allows it to achieve a more link load balanced network and is therefore better at avoiding congestion.

It is interesting to observe that the deployment of caches is not uniform for both of the topologies. This is due to the distribution of the NTs being non-uniform (see figure 2.6 and 2.8). One can also see that an area with a high population density also results in a higher density of caches in the same area. This is due to the fact that placing a cache at a CO results in a large amount of that CO's demand being handled without it needing to propagate across any network links.

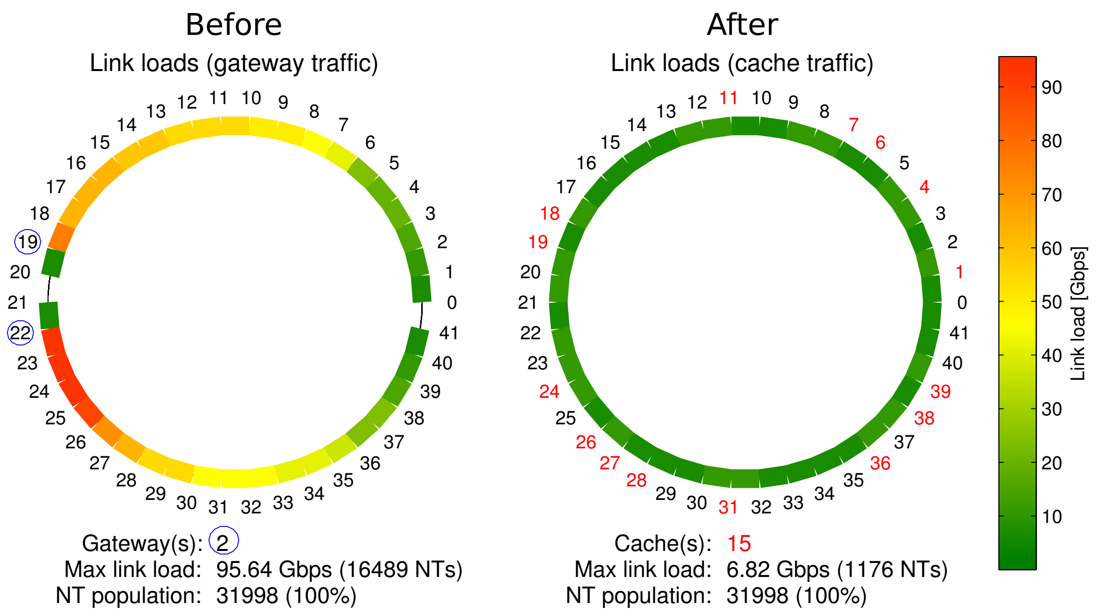


Figure 5.1: Best individual with 15 caches for the ring topology

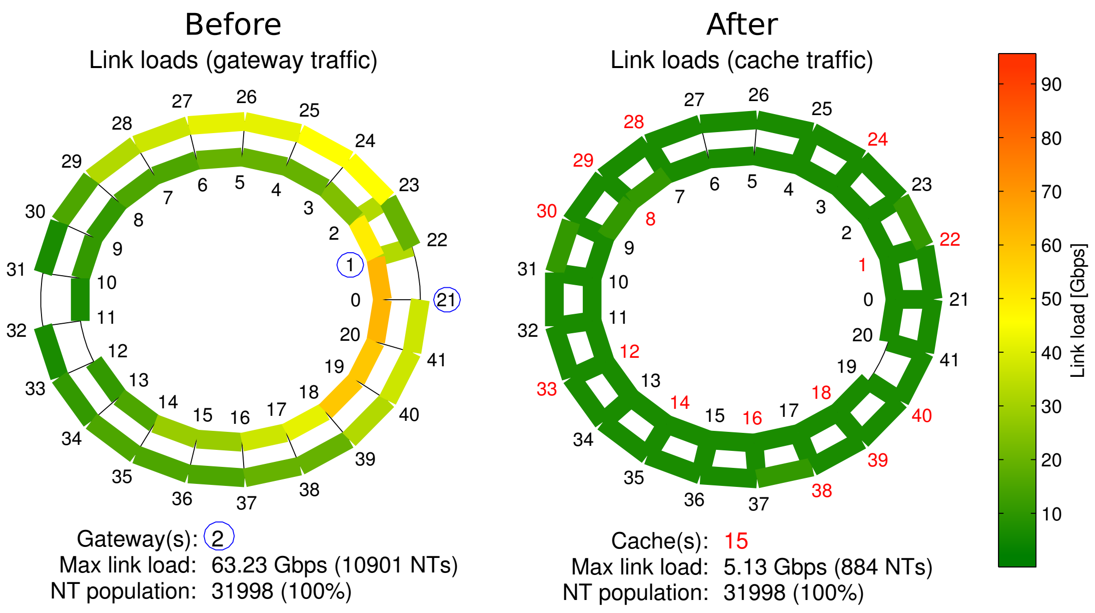

Before
Link loads (gateway traffic)
Gateway(s): 2
Max link load: 63.23 Gbps (10901 NTs)
NT population: 31998 (100%)
After
Link loads (cache traffic)
Cache(s): 15
Max link load: 5.13 Gbps (884 NTs)
NT population: 31998 (100%)
Link load [Gbps]

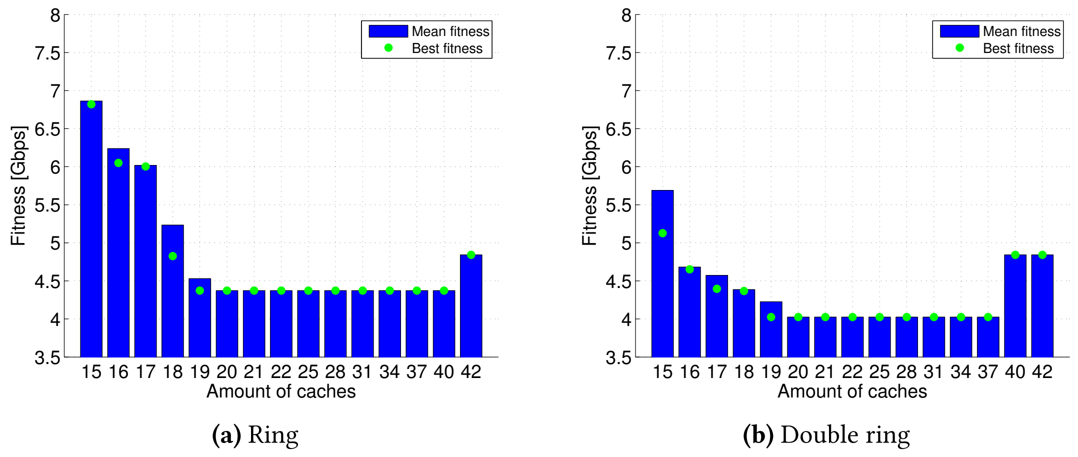
Figure 5.2: Best individual with 15 caches for the double ring topology

5.2 Cost-fitness trade-off analysis

A cost-fitness trade-off analysis is performed for both the ring and double ring topologies. Figure 5.3a and 5.3b shows the cost-fitness trade-off of the ring and double ring topologies respectively. As previously stated in section 4.2. The cache planner may choose the amount of caches in order to reach a specific maximum link load (robustness against congestion) with minimum required amount of caches.

Results are generated by performing 3 GA runs with different random seeds for each amount of caches. 3 runs is sufficient to find the best fitnesses and other good solutions. Few runs are chosen because of the long running times required to perform the trade-off analysis with the double ring taking 10.3 hours to complete. The amount of caches investigated are $15-22, 25, 28, 31, 34, 37, 40$ and 42. These amounts are chosen as to show the initial convergence towards a point where a better maximum link load can no longer be achieved and thereafter show how the fitness is non-decreasing (requires less sample points).

Both figures 5.3a and 5.3b show the same tendency. Starting at 15 caches with the highest maximum link load and progressively decreasing as expected until a point where adding additional caches will not decrease the maximum link load further and will not reach zero in maximum link load for 42 caches deployed either. The reason for the non-decreasing maximum link loads is because of the scenario's NT populations and the caches' capacities. A cache's capacity is 12 $Gbps$. If one looks at figure 2.6 of the NT populations for the ring, there are two COs (Ring: 6 & 26, Double ring: 18 & 29) that have a demand higher than 12 $Gbps$, being 16.61 $Gbps$ and 17.43 $Gbps$. Figures C.1 and C.2 in appendix C shows these demands. Due to these high demand occurring, it means that even if a cache is placed at these locations, there will always be at least $16.61 - 12 = 4.61$ *Gbps* and $17.43 - 12 = 5.43$ *Gbps* of demand occupying one or more links exiting from these two COs. *At least* means that due to the used routing algorithm, demand from other CO locations can occupy some of the cache capacity placed at these two high demand locations, which can result in more than the described remaining demand having to leave the CO.

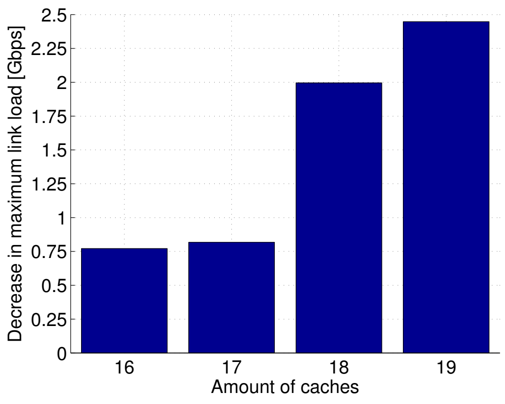


(a) Ring **(b)** Double ring

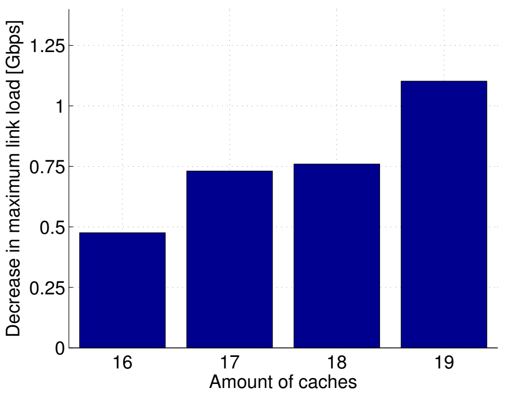
Figure 5.3: Cost-fitness trade-off of the two topologies

The reason for the increase in maximum link load with 42 caches in the ring topology and 40+42 caches in the double ring topology is simple. Due to the applied routing algorithm, there occurs different kinds of splits of the demand from the two COs with higher demand than a cache can handle. The reason for the higher maximum link load in the high amount of caches, is because the neighbouring caches have more available capacity, thereby allowing more traffic to occupy a link towards this shortest path cache. If one wanted to avoid this case, the modularity of the GA allows the planner to easily alter the routing algorithm. Such an alteration could possibly consist of a load balanced routing instead of a shortest path first routing algorithm. But one must keep in mind that when increasing the complexity of the routing algorithm, each evaluation of the objective function will increase in computation time. Therefore one must almost always perform an abstraction of the routing protocol's behaviour to reduce the computational load of each fitness evaluation and thereby also the running time of the GA.

The maximum link loads are achieved at 15 caches, with 6.82 *Gbps* for the ring topology and 5.13 *Gbps* for the double ring topology. The cheapest cost/amount of caches at which a minimum maximum link load is found, is by deploying 19 caches for either of the topologies, where the maximum link load is 4.02 *Gbps*. So an good choice of the amount of caches, that supplies the full demand of the COs is 15-19 caches. Another question is, what is the gain for each time a cache is added. Figures 5.4a and 5.4b show the decrease in maximum link load, when using the maximum link load from 15 caches as a baseline. This will allow one to clearly see the trade-off in cost/fitness for adding different amounts of caches. For example, in the ring topology it is seen that there is no significant difference in 16 and 17 caches, meaning that the cost of adding an additional cache may outweigh the small decrease in maximum link load. The same case is seen in the double ring topology when going from 17 to 18 caches.



(a) Ring



(b) Double ring

Figure 5.4: Decrease in maximum link load for different amounts of caches using 15 caches as a baseline.

5.3 Numerical evaluation

A numerical evaluation of the link loads, when content availability is considered, is performed as described in section 4.4. The amount of caches is symbolically chosen to be the minimum required amount of caches that satisfies all demand, 15. The gateway locations are as described in section 4.1.3, where two gateways are identified and placed in both network topologies. The content availabilities are also symbolically chosen to be 1, 0.9 and 0.8. Even though the case of content availability of 1 has been shown in the previous section, it is shown again to have comparable colour-schemes in link loads. Figures 5.5, 5.6 and 5.7 show the cache, gateway and total link loads for different content availabilities of the ring topology, while figures 5.8, 5.9 and 5.10 show them for the double ring topology. Using the assumption that ISPs utilise Gigabit Ethernet ports of 10, 40 or 100 $Gbps$, a discussion of link capacity planning is performed on the ring topology. No cross-traffic is also assumed for this discussion.

A content availability of 1 shows that the maximum link load is 6.82 $Gbps$, meaning that all links are below 10 $Gbps$. So for the case where all Netflix subscriber are interested in content available in caches deployed inside their ISP then there will be no need for increasing capacity in any links, when assuming they all have a standard link capacity of 10 $Gbps$.

In the case of a content availability of 0.9, there is a large difference in the maximum (total) link load with an increase of 7.11 $Gbps$ up to 13.93 $Gbps$, compared to a content availability of 1. Analysing the figure, shows that 8 links require a capacity of $10-17$ $Gbps$. This means that two 10 $Gbps$ ports for each of this links may be sufficient to handle the increased load, due to a fraction of the OTT content being serviced through gateways.

Lastly, the link loads associated with a content availability of 0.8 is analysed. The total link load for this case shows 28 links that require over 10 $Gbps$ links. For this content availability, a gradual increase of link load towards the gateways is more apparent, due to the gateways having a larger impact on the total link loads. This requires a gradual increase in link capacity, the closer to the gateways the links are. The maximum (total) link load, has increased to 22.06 $Gbps$ and is therefore potentially in the range where 40 $Gbps$ links are required dependent on the costs associated with it.

It is seen that smaller content availability is required before 100 $Gbps$ links are required, but when looking at figure 5.1 of the case of 0 content availability (all OTT traffic handled through gateways) for a ring topology, then it is definitely a requirement to have links with a capacity of 100 $Gbps$.

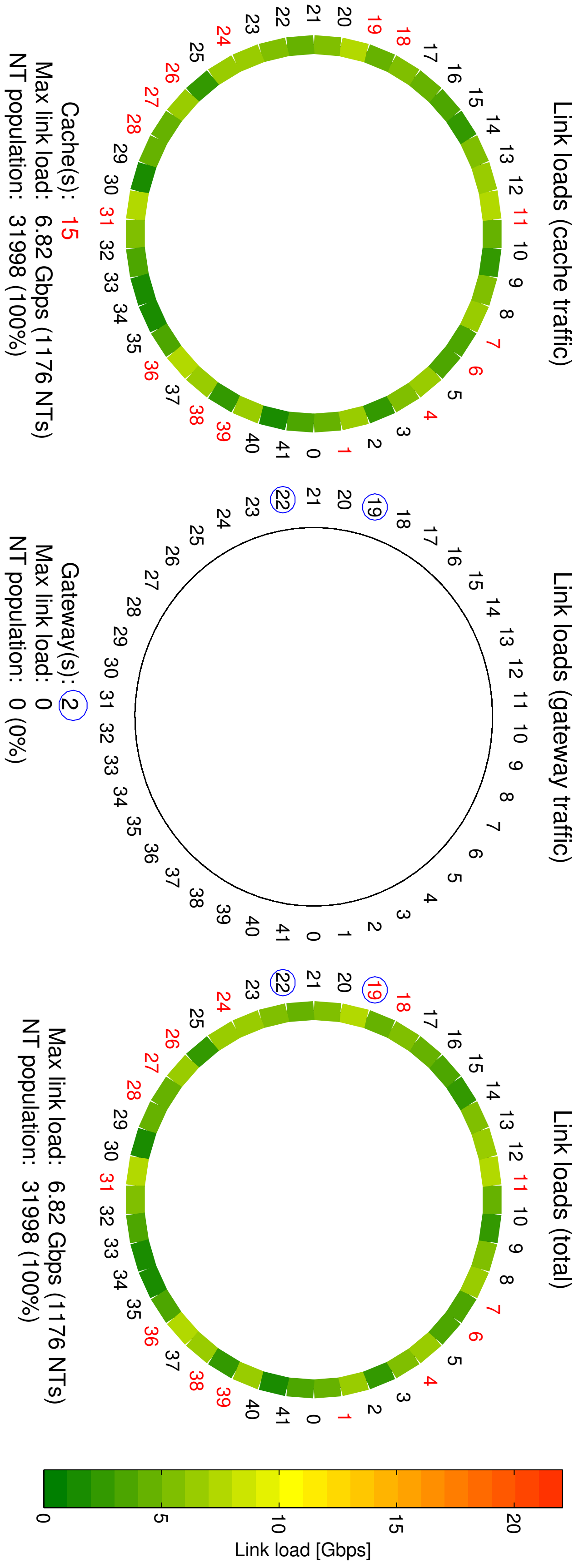


Figure 5.5: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 1

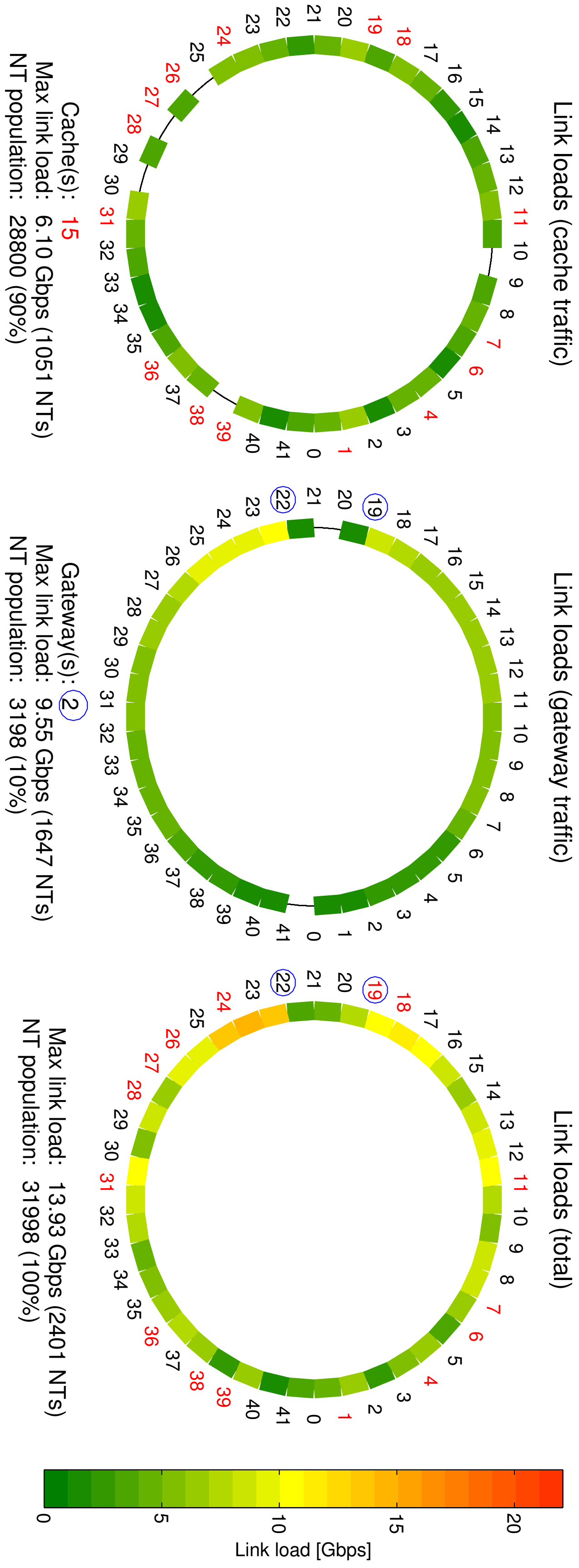


Figure 5.6: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 0.9

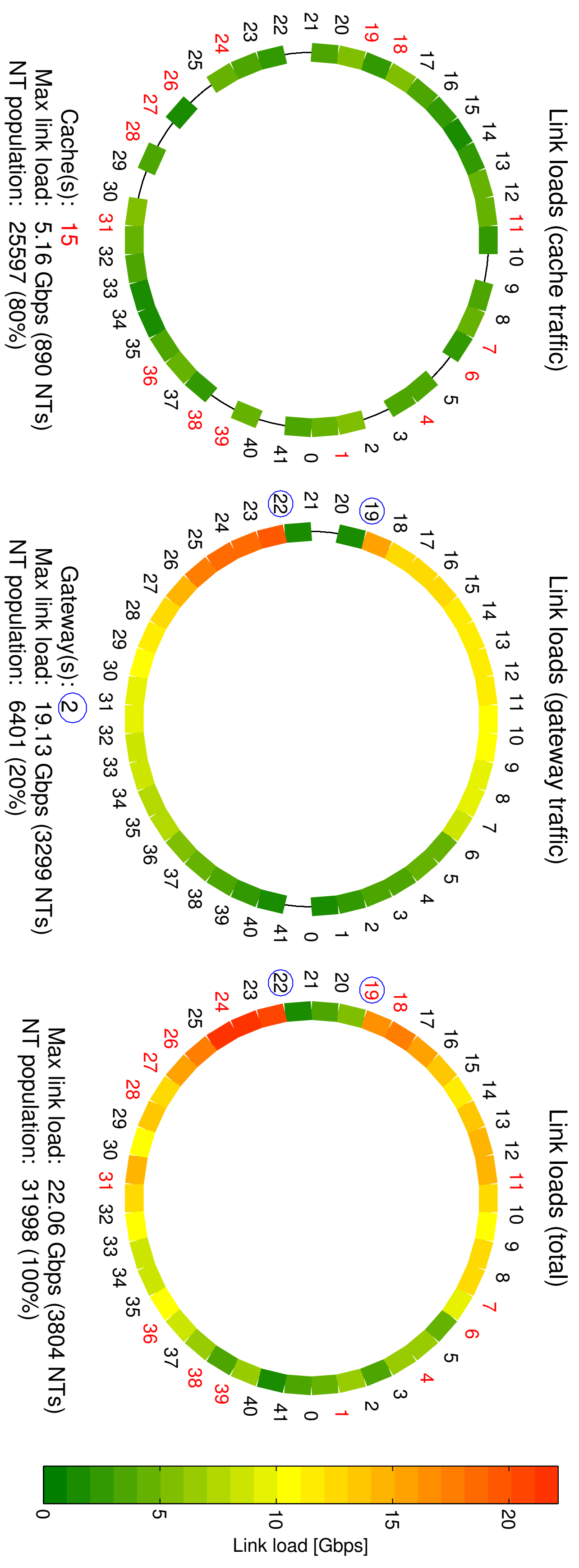


Figure 5.7: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 0.8

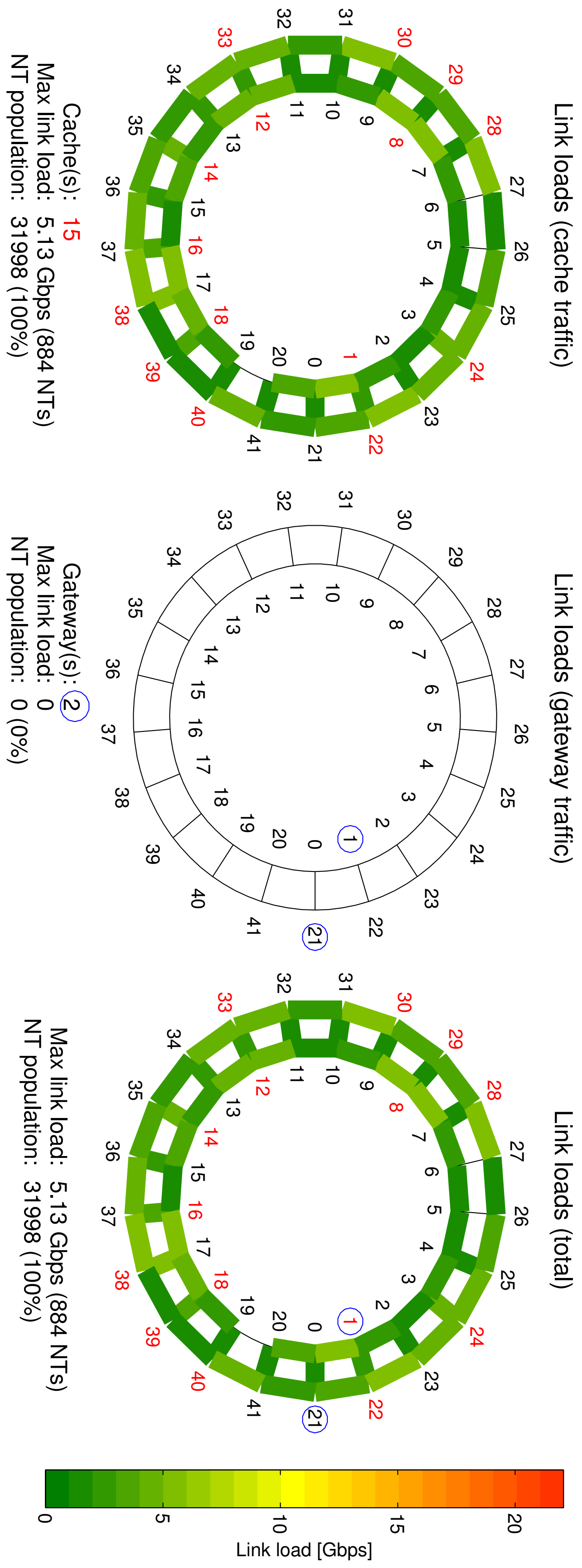


Figure 5.8: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 1

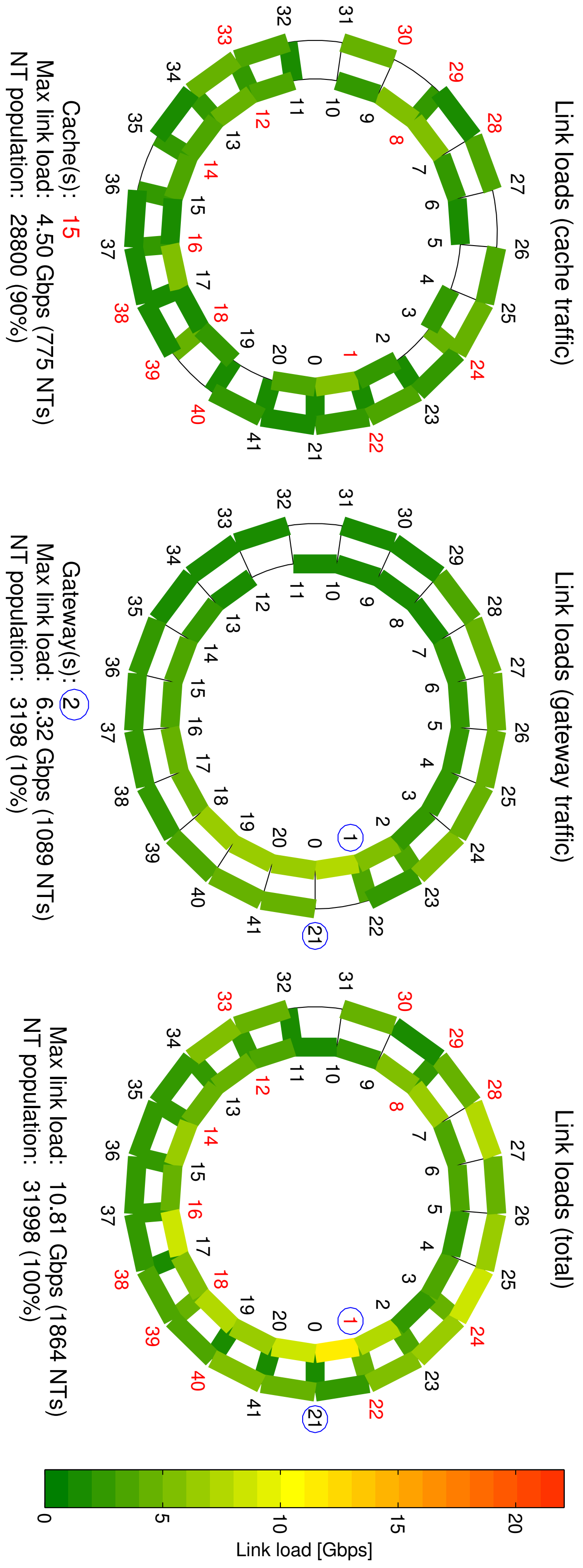


Figure 5.9: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 0.9

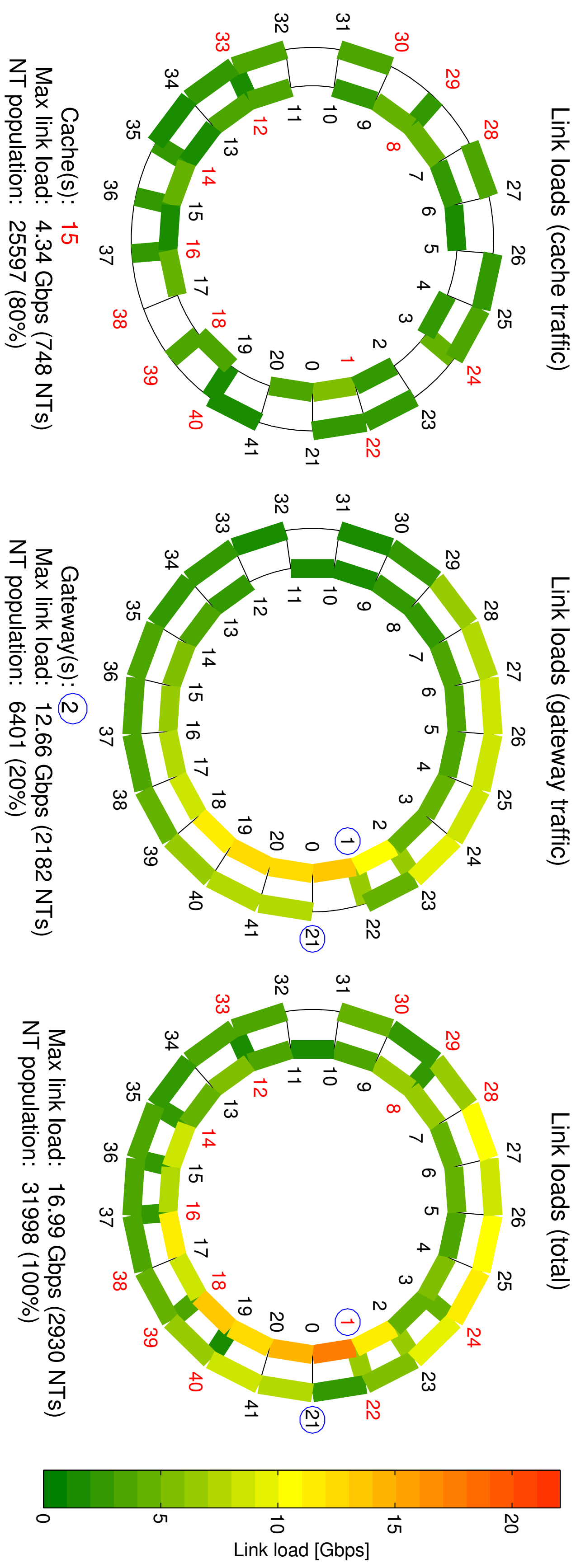


Figure 5.10: Link loads of the best found deployment of 15 caches with 42 CO locations using a content availability of 0.8

Chapter 6

Future work

In this chapter ideas for future work are presented. These ideas benefit the presented work in areas where delimitations are performed.

6.1 Analysing the affects of planning caches with a certain content availability

It is expected that when planning a cache deployment for a routing where content can be handled through both gateways and caches that a solution will only apply a good offload for a specific content availability. Therefore if drastic changes occur in user behaviour, the deployment of caches will no longer provide as good an offload. But it may still be beneficial to analyse how the link loads distribute across the network and how well this type of solution will perform under different content availability. For example, if a good solution is found for the case where 80% of the content is handled by caches and 20% is handled through gateways, how will the maximum link load change if 100% of the content is handled by caches. This would of course require the two plans to have the same amount of caches to be comparable in this sense. This will allow for an optimisation being performed for for both types of accessing content (cache and gateway).

6.2 Automatic estimation of parameter values for GA

The biggest limitation of this method is choosing the correct parameters that are good at solving the problem. If a fixed set of genetic operators and a range of parameters are known, a method for estimating a set of parameters for the GA would make this method much more user friendly and applicable in the real world. As it currently is for the implementation, a large amount of testing of different parameters and how they interact has resulted in many days of work, that could maybe have been performed more efficiently through some kind of machine learning. Even a GA could solve this type of problem, but may require a large amount of computation time, as a single fitness evaluation would consists of a complete run of a GA.

6.3 Planning for a subset of the population

The current implementation of the method works with the idea of planning for the complete set of an areas population, and satisfying the complete demand of the network. If one had data on a correct or estimated proportion of Netflix subscribers or estimated maximum number of Netflix subscribers, one could plan for this amount of demand. If one had a geographical analysis of different target audiences of e.g. age groups that live in certain areas, one could estimate Netflix subscribers if knowledge about Netflix user age groups is known. There are countless factors within GIS that could affect the demand from each CO.

6.4 Apply the method to an actual ISP's network

By applying this method on an actual ISP's networking infrastructure with its given adjacencies, solutions can be found that realistically portray link loads. This would also allow one to learn how well the algorithm performs under more complex network infrastructures.

6.5 Take cross-traffic into account

As part of the numerical evaluation of link loads, when content availability is taken into account, cross-traffic can also be considered. This will allow the ISP's network link capacity planner to more accurately predict where and when potential congestions can occur and be aware of these before they happen. The planner can then avoid these congestions by dimensioning the link capacities accordingly.

Chapter 7

Conclusion

As it was stated in the introduction, over-the-top (OTT) content has become increasingly popular, accounting for 67.4% of downstream traffic at peak periods. This has resulted in OTT content providers such as Netflix and Google to deploy caches in the network to offload the ISP's backhaul links and provider a better service. This is the focus of this thesis and resulted in the following problem statement:

> "When ISP networks display a large amount of OTT network traffic and exhibit congested links, how can caches be planned in an ISP's network to offload the backhaul network and provide a better QoS for OTT content subscribers?"

To first tackle this problem, the field of study had to be identified. The field of study was narrowed down to facility location (FL). These are NP-hard combinatorial optimisation problems, and not easily solvable. The problem of this thesis is defined as placing a set of capacity constrained caches at potential locations given by COs. This is within the sub-category of FL, namely capacitated facility location (CFL), since the facilities are capacity constrained. This thesis focuses on solving the problem of finding the most link load balanced network through a minimisation of the maximum link load. This is a undefined problem in literature, which resulted in the Link Load Balanced Capacitated Facility Location (LLB-CFL) problem being defined. To solve this problem, existing methods for solving FL problems were investigated. This analysis resulted in the choice of a metaheuristic, namely Genetic Algorithms (GAs), as the optimisation algorithm. This choice was performed, since it had shown good results in solving FL problems, is broadly applicable, conceptually simple and can provide good local optimal solutions within feasible running times. Other more exact solutions, such as branch and bound (linear programming), but require impractical running times. GAs use principals of natural evolution to perform a search through a solution space.

The choice of GAs, resulted in a analysis of its complete process and various techniques for the following GA processes:

- Initial population
- Objective and fitness function
- Selection
- Crossover and mutation
- Visualisation of evolution
- Termination criterion

It was then necessary to find which operators were applicable to the problem and which were not. This separation was performed through the constraint to the amount of caches that were allowed to be deployed. Since many mutation and crossover method do not provide this functionality, a crossover function was identified through literature and a probability based mutation function was proposed.

In order to find the demand and constraints to the caches and how the caches should act, a case study was performed on Netflix. This resulted in demand being defined as one NT demanding a Super HD video stream of 5800 $Kbps$ and a constraint to the caches of 12 $Gbps$ given by a caches sustainable throughput. The caches were also defined as directed caches, meaning that they could only receive requests and handle them. They do not have features allowing them to decide where demand should be serviced from and can not relay requests to other caches either. This means that an NT can only connect to caches with sufficient capacity to handle it. The thesis focuses on a worst-case for the link loads in the network, where all NTs are streaming OTT content from caches at the same time.

As an input to the proposed method, GIS data is utilised. This includes data on CO locations, their associated NT populations and their adjacency. Two different adjacency matrices are constructed based on two different topologies; ring and double ring. This will allow the method to be applied on two different topologies that are compared in the results. An output is defined as the set of locations placed at a set of these CO locations. From these locations paths between COs and caches are constructed using a routing algorithm that also provides associated costs to each path.

A big part of the proposed method is identifying what works well and how the problem can be defined in the domain of GAs. This involves identifying which techniques are applicable, are good at finding link load balanced solutions and have a good efficiency. This is performed through an analysis of different parameters for each of the genetic operators. A method for cost-fitness trade-off analysis is also proposed, involving defining a range of the amount of caches that need to be investigated. The trade-off between cost and fitness must then be analysed to perform a choice on the most cost effective solution that still upholds some requirements to a maximum link load. A method for numerical analysis of the link loads, when a percentage of the traffic is handled through gateways, is also defined. This involves defining a specific content availability, defining how much of the demand is handled by caches internally in the ISP's network. The change in link loads and maximum link loads can then be investigated to gain knowledge on potential areas of congestion.

Advantages and disadvantages are also defined. Among the advantages are a short running time that provides a good link load balanced solution. Another big advantages is how easy the implementation and method in general is to be applicable to solve other problems that seek to find a good deployment of servers that provide a link load balanced network. A last advantage is the modularity of the method, inherited from the GA, where each process can be interchanged as long as it upholds the constraints of the problem definition. Among the disadvantages or limitations is that the method can not guarantee that the found solution is globally optimal. It does on the other hand provide very good fitness solutions within a reasonable amount of time. The method also requires a large amount of parameter tweaking, meaning that if the characteristics of the problem changes too much, the parameters may need to be changed as well. This may specifically be the case for altering or changing the routing algorithm, which greatly affects the result.

The results shows that the initial problem statement is fulfilled. A method is proposed that finds a good solution for deploying caches in an ISP's network that offloads the backhaul of the network. In the case of the ring topology a maximum link load is reduced from 95.64 $Gbps$ to 6.82 $Gbps$ by deploying 15 caches in the network. For the double ring topology the maximum link load is reduced from 63.23 $Gbps$ to 5.13 $Gbps$. This is due to its increased number of links, allowing for a better spread of the load. A better QoS is achieved through the minimisation of the maximum link load, allowing for an increased robustness against unpredictable traffic growths and avoidance of congestion. This allows for the bitrate of Super HD streams to be unaltered during cross-traffic that propagates along the same links. A cost-fitness trade-off analysis of both topologies has shown that an identification of when an additional cache will provide a sufficient reduction in maximum link load is worth the costs. The numerical analysis of link loads when a percentage of the demand is handled through gateways also identifies the links on which potential congestion can occur if a certain link capacity is unavailable.

To summarise, this thesis has been an educational experience with an analyses, design and implementation of a method for planning caches in an ISP's network. This includes gaining invaluable practical experience with solving a NP-hard combinatorial problem, specifically facility location (FL), using Genetic Algorithms (GAs). This thesis was an overall great experience and one can only hope that the knowledge gained and documented can be useful for deploying caches in practice.

Bibliography

Correa, E. S., Teresinha, M., Steiner, A., Freitas, A. A., Parana, C. D., and Carnieri, C. (2001). A genetic algorithm for the p-median problem.

Empson, R. (2013). Netflix tops hbo in paid u.s. subscribers as members stream 5 billion hours of content in q3. Online: `http://techcrunch.com/2013/10/21/netflix-tops-hbo-in-paid-u-s-subscribers-as-members-stream-5-billion-hours-of-content-in-q3/`, Retrieved March 13th 2014, attached as PDF - Netflix Tops HBO In Paid U.S.pdf.

Evers, J. (2013). Netflix prime time isp performance metrics. blog.netflix.com. Online: `http://blog.netflix.com/2013/11/new-netflix-prime-time-isp-performance.html`, Retrieved March 11th 2014, attached as PDF - New-Netflix-Prime-Time-ISP-Performance-Metrics.pdf.

Farahani, R. and Hekmatfar, M. (2009). *Facility Location: Concepts, Models, Algorithms and Case Studies*. Contributions to Management Science. Physica.

GEATbx (2014). Selection. Online: `http://www.pg.gda.pl/~mkwies/dyd/geadocu/algselct.html`, Retrieved May 15th 2014, attached as PDF - Bias-Spread.pdf.

Gong, D., Gen, M., Yamazaki, G., and Xu, W. (1997). Hybrid evolutionary method for capacitated location-allocation problem. *Computers & industrial engineering*, 33(3):577–580.

Holland, J. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105.

Labovitz, C. (2012). First data on changing netflix and cdn market share. `http://www.deepfield.com/2012/06/first-data-on-changing-netflix-and-cdn-market-share`.

Li, X. and Yeh, A. G. (2005). Integration of genetic algorithms and gis for optimal location search. *International Journal of Geographical Information Science*, 19(5):581–601.

Matlab (2014a). Fitness scaling. Online: `http://www.mathworks.se/help/gads/fitness-scaling.html`, Retrieved May 14th 2014, attached as PDF - Fitness-scaling.pdf.

Matlab (2014b). Global optimization toolbox. Online: `http://www.mathworks.se/products/global-optimization/`.

Netflix (2013a). Highest quality hd now available to all netflix members. Online: `http://blog.netflix.com/2013/09/highest-quality-hd-now-`

available-to-all.html, Retrieved March 16th 2014, attached as PDF - highest-quality-hd-now-available-to-all.pdf.

Netflix (2013b). Netflix open connect content delivery network. Online: http://oc.nflxvideo.net/docs/OpenConnect-Deployment-Guide.pdf, Retrieved March 13th 2014, attached as PDF - OpenConnect-Deployment-GuideNetwork.pdf.

Netflix (2014). Netflix open connect content delivery network. Online: https://signup.netflix.com/openconnect, Retrieved March 13th 2014, attached as PDF - Netflix-Open-Connect-Content-Delivery-Network.pdf.

Rayburn, D. (2010). Akamai to lose netflix as a customer, level 3 and limelight pick up the business. Online: http://blog.streamingmedia.com/2010/11/akamai-to-lose-netflix-as-a-customer-level-3-and-limelight-pick-up-the-business.html, Retrieved March 13th 2014, attached as PDF - Akamai-To-Lose-Netflix-As-A-Customer-Level-3-and-Limelight-Pick-Up-The-Business.pdf.

Rayburn, D. (2014). Here's how the comcast & netflix deal is structured, with data & numbers. Online: http://blog.streamingmedia.com/2014/02/heres-comcast-netflix-deal-structured-numbers.html, Retrieved March 13th 2014, attached as PDF - Heres-How-The-Comcast-and-Netflix-Deal-Is-Structured-With-Data-and-Numbers.pdf.

Sandvine (2013). Global internet phenomena report 2h 2013. Online: https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/2h-2013-global-internet-phenomena-report.pdf, Retrieved March 13th 2014, attached as PDF - 2h-2013-global-internet-phenomena-report.pdf.

Sandvine (2014). Global internet phenomena report. Online: https://www.sandvine.com/trends/global-internet-phenomena/.

Santos, D., De Sousa, A., Alvelos, F., Dzida, M., Pióro, M., and Zagozdzon, M. (2009). Traffic engineering of multiple spanning tree routing networks: the load balancing case. In *Next Generation Internet Networks, 2009. NGI'09*, pages 1–8. IEEE.

Sasaki, S., Comber, A. J., Suzuki, H., and Brunsdon, C. (2010). Using genetic algorithms to optimise current and future health planning-the example of ambulance locations. *International journal of health geographics*, 9(1):4.

Sastry, K., Goldberg, D., and Kendall, G. (2005). Genetic algorithms.

Sivanandam, S. N. and Deepa, S. N. (2007). *Introduction to Genetic Algorithms*. Springer Publishing Company, Incorporated, 1st edition.

Spangler, T. (2013). Twc: Netflix is withholding content to gain 'unprecedented' access to isps. Online: http://www.multichannel.com/cable-operators/twc-netflix-withholding-content-gain-unprecedented-access-isps/141261, Retrieved March 16th 2014, attached as PDF - twc-netflix-withholding-content-gain-unprecedented-access-isps.pdf.

TeleGeography (2014). Submarine cable map. Online: http://www.submarinecablemap.com/.

Thorkildsen, J. (2014a). Netflix go to bed with the enemy and challenging the net neutrality. what happens? (translated from norwegian). Online: http://filtermagasin.no/netflix-demonterer-internett-slik-vi-kjenner-det-pastar-nettkjenner-hva-er-det-som-skjer/, Retrieved March 16th 2014, attached as PDF - netflix-comcast.pdf.

Thorkildsen, J. (2014b). Netflix signs agreement with telenor, consumer council is concerned with net neutrality (translated from norwegian). Online: http://filtermagasin.no/netflix-inngar-avtale-med-telenor-forbrukerradet-er-bekymret-for-nettnoytraliteten/, Retrieved March 16th 2014, attached as PDF - netflix-telenor.pdf.

Youtube (2014). Statistics - youtube. Online: http://www.youtube.com/yt/press/statistics.html, Retrieved March 18th 2014, attached as PDF - Statistics-YouTube.pdf.

Appendix A

Pseudo-code for mutation by hypermutation

```
Step 1.
Randomly select a subset of 10% of the individuals from the entire
    population.
Step 2.
FOR EACH individual X selected in step 1 DO
  Let H be the set of facility indices that are not currently present in
      the genotype of
individual X
  FOR EACH facility index i included in set H DO
    BEST = X
    FOR EACH facility index j that is currently present in the genotype of
        the individual X DO
      Let Y be a new individual with the set of facilities given by:
      Y = (X − {j}) ∪ {i}
      Calculate the fitness of Y
      IF fitness(Y ) < fitness(BEST) THEN
        BEST = Y
      END IF
    END FOR
    IF fitness(BEST) < fitness(X) THEN
      X = BEST
    END IF
  END FOR
  Insert the new X into the population, replacing the old X
END FOR
```

Algorithm A.1: Heuristic hypermutation [Correa et al., 2001]

Appendix B

Pseudo-code for objective function

Algorithm 4 Objective function

```
S: Individual / cache deployments
n: Amount of COs
N: NT population vector (demand of each CO)
A: Adjacency matrix
X,Y: xy-coordinate vectors
PopFrac: Fraction of NT population vector to add at each loop
Bandwidth: Bandwidth of OTT content traffic from each NT
CacheCap: Maximum sustainable throughput from a cache

function ObjectiveFunction($S, n, N, A, X, Y, PopFraction, Bandwidth, CacheCap$)
    - Find all paths and their OD distance from all COs to each deployed cache
    $P, C = PathsAndDist(A, X, Y)$
    - Associate NTs to caches using round robin and shortest path first
    $Assoc, CacheLoads = Associate(n, N, PopFrac, Bandwidth, CacheCap)$
    - Find load matrix describing cache and link loads
    $L = CalcLoadMatrix(n, P, Assoc, CacheLoads)$
    return $L$
end function
```

Algorithm 5 Shortest paths and distances

```
function PATHSANDDIST($A, X, Y$)
    for each CO, $i$ do
        for each deployed cache, $j$ do
            $P(i, j) \leftarrow ShortestPath(A, X, Y, i, j)$        ▷ Construct matrix of path lists
            $C(i, j) \leftarrow TotalDist(C, P(i, j))$        ▷ Construct cost matrix of OD distances
        end for
    end for
    return $P, C$
end function
```

Algorithm 6 Associate NTs to caches

```
1: function ASSOCIATE($N, PopFrac, Bandwidth, CacheCap$)
2:     $NFrac \leftarrow round(N \cdot PopFrac)$
3:     $Assoc \leftarrow zeros(n,n)$          ▷ Amount of demand from CO associated to cache
4:     $CacheLoads \leftarrow zeros(n)$
5:     $running \leftarrow True$
6:     while running do
7:         for each CO, $i$ do
8:             if N(I) not empty then
9:                 $CID \leftarrow sort(C(i,:))$          ▷ Sort cache IDs by SP distance from CO $i$
10:                for each cache, $j$, in CID do
11:                    if $CacheLoad(CID(i)) + NFrac(i) \leq \frac{CacheCap}{Bandwidth}$ then
12:                        if N(i) > NFrac(i) then
13:                            $amount = NFrac(i)$
14:                        else
15:                            $amount = 1$
16:                        end if
17:                        $CacheLoads(CID(j)) \leftarrow CacheLoads(CID(j)) + amount$
18:                        $Assoc(i,CID(j)) \leftarrow Assoc(i,CID(j)) + amount$
19:                        $N \leftarrow N - amount$
20:                        $Break\ for\ loop$
21:                    else if $CacheLoad(CID(i)) + 1 \leq \frac{CacheCap}{Bandwidth}$ then
22:                        $CacheLoads(CID(j)) \leftarrow CacheLoads(CID(j)) + 1$
23:                        $Assoc(i,CID(j)) \leftarrow Assoc(i,CID(j)) + 1$
24:                        $N \leftarrow N - 1$
25:                        $Break\ for\ loop$
26:                    end if
27:                end for
28:            end if
29:        end for
30:        if $\sum N == 0$ then
31:            $running \leftarrow False$
32:        end if
33:    end while
34:    return $Assoc, CacheLoads$
35: end function
```

Algorithm 7 Calculate load matrix

```
 1: function CalcLoadMatrix($n$, $P$, $Assoc$, $CacheLoads$)
 2:     $L = zeros(n, n)$
 3:     $L = L + diag(CacheLoads)$
 4:     for each CO, $i$ do
 5:         for each cache, $j$ do
 6:             if Cache, $j$, is deployed then
 7:                 for Each hop, $k$, in the path P(i,j)[1:end-1] do
 8:                     $L(P(i,j)[k], P(i,j)[k+1]) = L(P(i,j)[k], P(i,j)[k+1]) + Assoc(i,j)$
 9:                 end for
10:             end if
11:         end for
12:     end for
13: end function
```

Appendix C

Demand from COs in Gbps

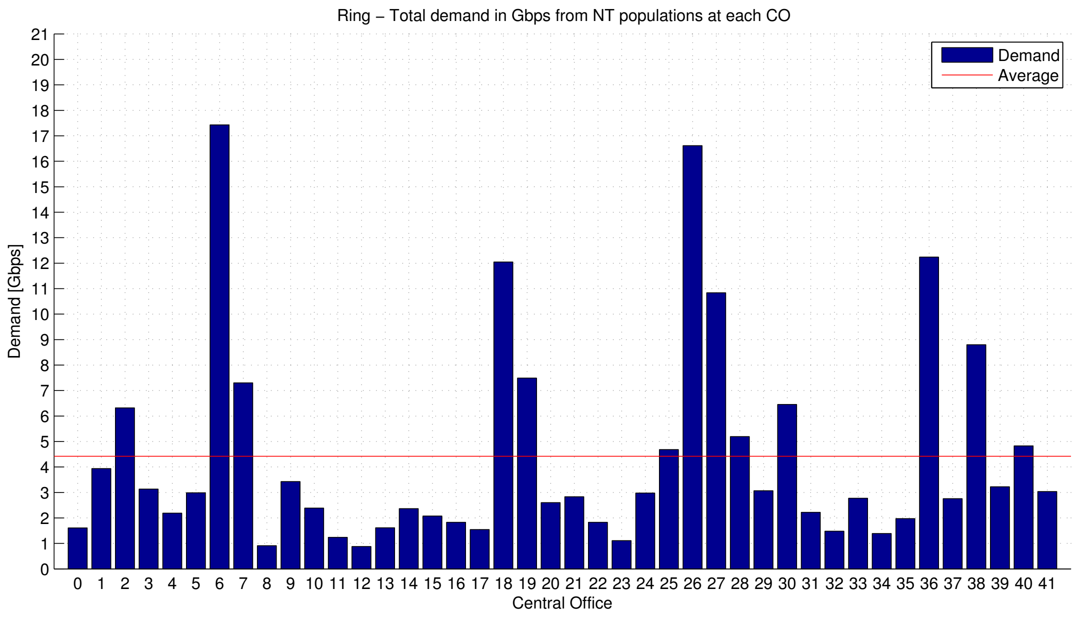


Figure C.1: Ring - Total demand in Gbps from NT populations at each CO

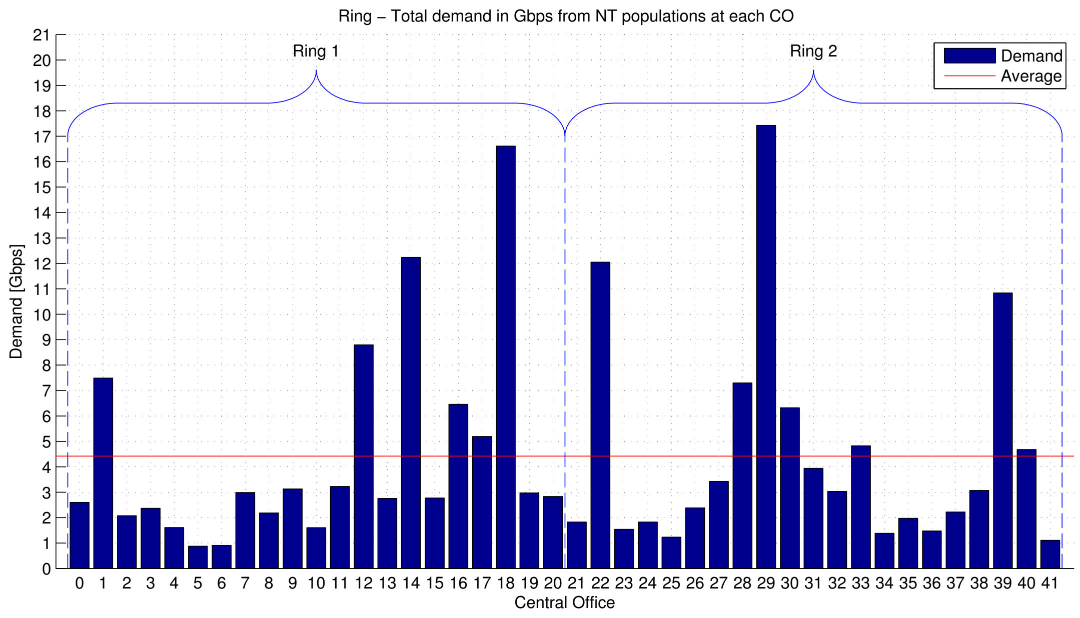


Figure C.2: Double ring - Total demand in Gbps from NT populations at each CO

Appendix D

Cache loads of minimum required caches

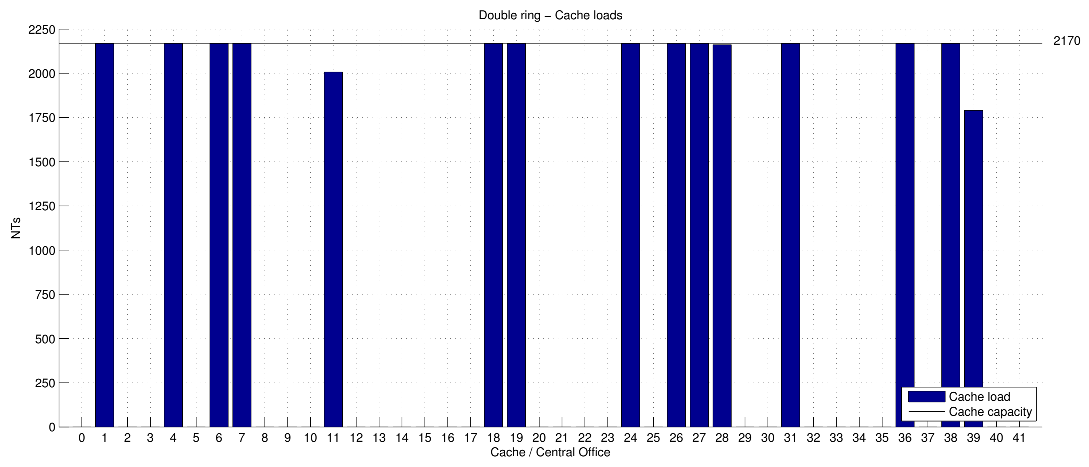


Figure D.1: Cache loads of best found deployment of 15 caches for ring topology

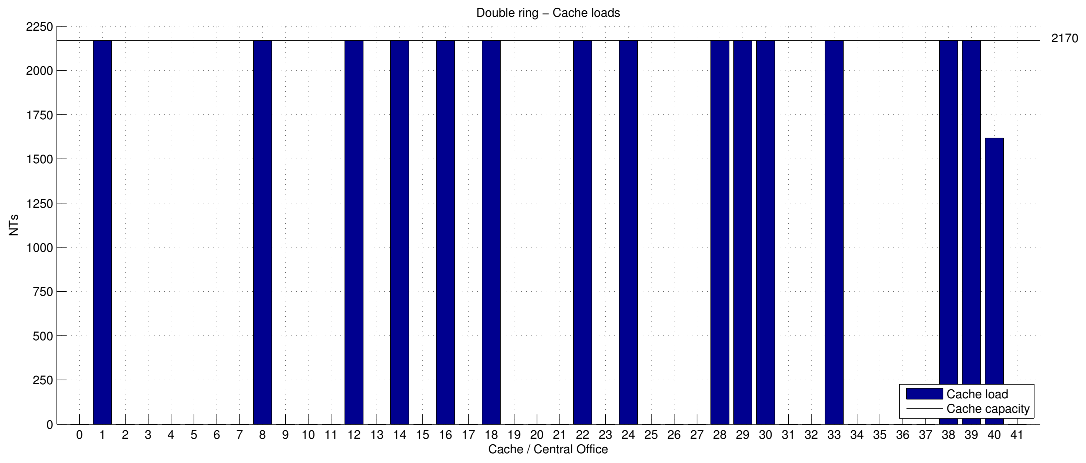


Figure D.2: Cache loads of best found deployment of 15 caches for double ring topology

Appendix E

Used acronyms

CAPEX capital expenditure

CDN content delivery network

CFL capacitated facility location

CO central office

FL facility location

GA Genetic Algorithm

GIS Geographical Information System

ISP Internet service provider

IX Internet Exchange

NT network termination

OPEX operational expenditure

OSPF Open Shortest Path First

OTT over-the-top

LLB-CFL Link Load Balanced Capacitated Facility Location

QoS quality of service

RTE real-time entertainment

SP segment point

BGP Border Gateway Protocol