CHALMERS
UNIVERSITY OF TECHNOLOGY

Algorithms for Reducing Fragmentation on a Radio Base Station Transport Link Using CPRI

Isac Andersson

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

MASTER THESIS 2020: EENX30

Algorithms for Reducing Fragmentation on a Radio Base Station Transport Link Using CPRI

Isac Andersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communications, Antennas and Optical Networks
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Algorithms for Reducing Fragmentation on a Radio Base Station Transport Link Using CPRI
Isac Andersson

Supervisor: Mohammad H. Moghaddam, Department of Electrical Engineering
Examiner: Jiajia Chen, Department of Electrical Engineering

Master Thesis 2020: EENX30
Department of Electrical Engineering
Division of Communications, Antennas and Optical Networks
Chalmers University of Technology
SE-412 58 Gothenburg
Phone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Abstract

The increasing demands for the next generation of cellular networks impose big challenges for radio equipment manufacturers. A critical part in order to meet these demands is to reduce the required interconnect resources within the radio base stations (RBS), for which CPRI is the dominating standard. In this thesis, potential algorithms for reducing fragmentation on CPRI links were investigated. In particular, the advantages of using offline processing compared to online implementations for the resource allocation were examined. A simulation environment in which RBS configurations were modelled was designed and implemented in order to evaluate the algorithms. Based on the simulation results, it was shown that there is more potential for improvement in complex RBS configurations, with many radio units and mixed line bit rates. The best offline algorithm managed to allocate up to 15 % more slots on average than the best online alternative. The results were considered promising, and it was recommended to investigate them further for more specific use cases.

Acknowledgements

I would like to thank the great people at Ericsson who have helped me throughout this project. In particular, I would like to thank Peter Karlsson, who has shown great interest in the topic and contributed with his expertise and guidance. Thanks also to Martin Åhsberg for the opportunity to do the thesis, to Mohammad H. Moghaddam, and to Team Schrödinger and LC5 for their support.

Isac Anderssson, Gothenburg, May 2020

Abbreviations

AxC	Antenna-Carrier
BTS	Base Station
CFF	Combined First-Fit Algorithm
CPRI	Common Public Radio Interface
eCPRI	Ethernet-Based CPRI
C-RAN	Centralized Radio Access Network
FC	Favourable Changes Algorithm
HFF	High First-Fit Algorithm
IQ	In-Phase and Quadrature
LFF	Low First-Fit Algorithm
MF	Minimal-Format Algorithm
NP	Non-Deterministic Polynomial
OPT	Optimal algorithm (in terms of minimizing subframes)
RAN	Radio Access Network
RB	Region-Based Algorithm
RE	Radio Equipment
REC	Radio Equipment Controller
RBS	Radio Base Station
RF	Radio-Frequency
SA	Sorting Algorithm

Contents

List of Figures

List of Tables

1

Introduction

The launching of 5G imposes big challenges for radio equipment manufacturers. A critical part of the radio access networks (RAN) is the internal communication between baseband and radio units within radio base stations (RBS), referred to as the fronthaul network. The Common Public Radio Interface (CPRI) was developed to replace the overhauled coaxial-based systems used in legacy fronthaul networks to allow for using fiber-optic infrastructure and, subsequently, enabled meeting the demands of a rapidly increasing mobile data traffic. With the next generation of mobile networks currently being launched, telecommunication systems needs to support 1000 times higher mobile data volume per area, 10 to 100 times higher number of connected devices and 10 to 100 times higher typical user data with respect to 4G LTE [1]. Ongoing research evaluates whether CPRI will be able to manage such changes [1][2][3]. In [4], CPRI is called the bottleneck of today's RANs. In addition, energy consumption for radio networks is becoming more significant, both from a cost and environmental aspect [5], which puts demands on optimizing the resources.

An RBS configuration can consist of multiple baseband and radio units. For large configurations, using the interconnect resources optimally is complicated. The aim of this thesis is to investigate allocation algorithms for reducing fragmentation on CPRI links, in order to use radio interconnect resources more efficiently.

1.1 Background

To describe the problem, some basic keywords and concepts needs to be introduced, including the evolution of RAN, the CPRI standard and the allocation of interconnect resources using CPRI. Interested readers can find more information about previous RAN architecture in [6]. For deeper knowledge on CPRI, and especially details concerning the physical layer, the CPRI specification [7] is recommended, as well as [3].

1.1.1 Radio Access Networks

A RAN is responsible for connecting user equipment to the backhaul network. Traditionally, stand-alone base stations (BTS) constituted a RAN, each operating independently. All functionalities of radio equipment (RE) and radio equipment controllers (REC) were usually built in the BTS and housed in a single cabinet, connected to an antenna in a BTS tower. This setup had several flaws, including problems dealing with fluctuating traffic, coverage and network performance and insufficient capacity [6]. As 3G was launched, the REC and RE was separated, introducing the distributed base station architecture. This allowed for placing the RE closer to the antenna, reducing losses related to the radio-frequency (RF) signal transmission between the antenna and RU. The communication

standards Open Base Station Architecture Initiative (OBSAI) and Common Public Radio Interface (CPRI) [7] was established for the transport link between the REC and RE, enabling it to be used more efficiently with fiber. In this report, only fronthaul networks using CPRI is considered, as CPRI has become the dominating standard in the telecommunication industry [1][3][6]. The distributed BTS architecture developed further into Centralized-RAN (C-RAN) where multiple RECs are centralized at one location (see Fig. 1.1.1), and Virtualized-RAN (V-RAN), with RECs virtualized in the cloud. Centralized RECs allows for more efficient solutions, as different RECs can cooperate to handle traffic fluctuation, avoiding interference among BTS units and more [3][8]. The later allows for dense placement of REs. In addition, connecting multiple REs to a REC pool facilitates the installation part [9]. With new functionalities, the costs for network operators can be reduced and RAN architectures gets more scalable [8].

As the evolution of RAN technologies has moved forward, the topology of an RBS is no longer just a stand-alone BTS. With C-RAN, an RBS can consist of multiple RECs and REs, resulting in more complex configurations which put constraints on the internal communication links within the RBS.

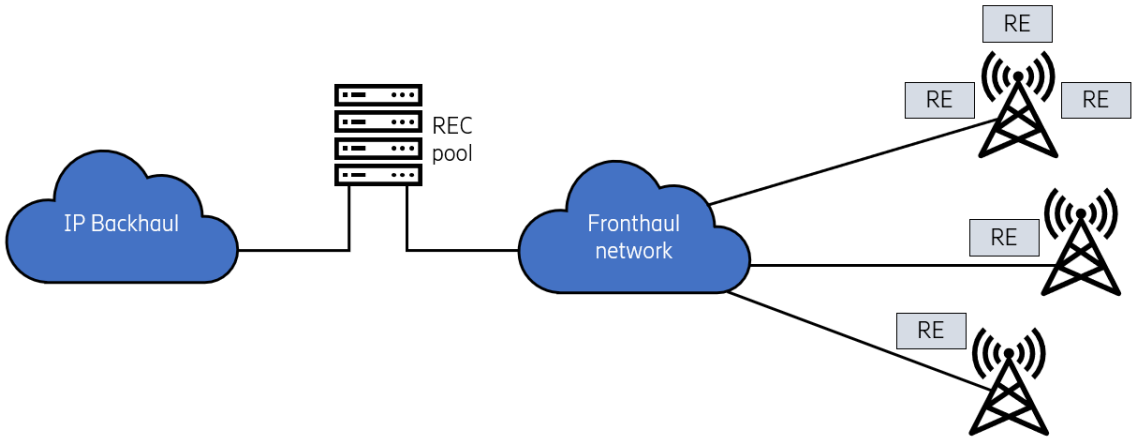


Figure 1.1.1: *Simple outline of a C-RAN network architecture.*

1.1.2 CPRI Transport Concept

CPRI specifies layer 1 & 2 protocols for digitized internal interfaces between the RE and REC within an RBS. It uses serial communication and transmits over a dedicated channel with constant bit rate. Different CPRI links can support line bit rates from 0,614 to 24,330 Gbps. Data is transferred using four different flows; control plane, management plane, synchronization and user plane. This report concerns resource allocation for the user data plane, which allows for two-way communication of data between the RE and REC. Fig. 1.1.2 shows the CPRI transport concept, where incoming radio-frequency (RF) data is received and digitized by the RE before being transported as IQ-data (in-phase and quadrature) over the CPRI link to the REC, and later on to the IP backhaul. Similarly, data entering the REC from the IP backhaul is transported over the CPRI link through the user data signal flow on to the RE, converted to an analogue signal before being radiated in the air interface.

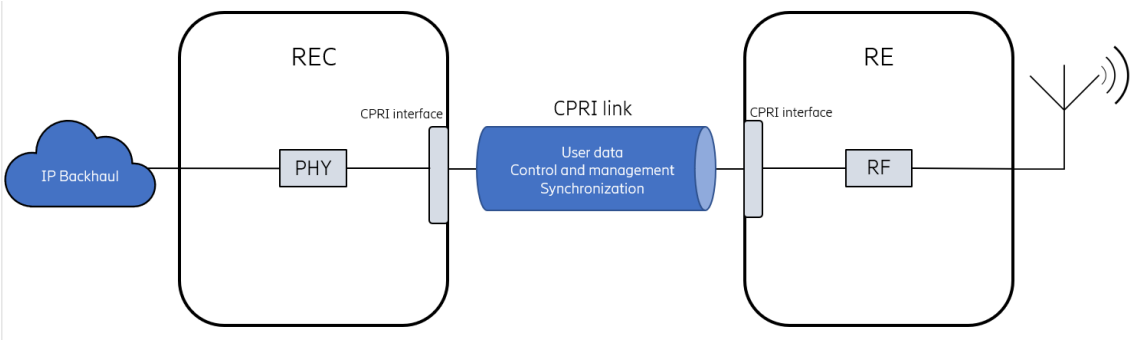


Figure 1.1.2: *Illustration of the CPRI transport concept between an REC and an RE.*

Antennas are configured to serve different sectors, each corresponding to an antenna carrier-group (AxC-group). The structure of an AxC-group is shown in Fig. 1.1.3. A carrier branch (also referred to as just 'carrier' in this thesis) carries IQ-data and is central for transmission in the CPRI user data plane. Carrier branches belonging to the same antenna group shares the same attribute values, such as sample rate & width, destination and radio frame length. This is referred to as a carrier set in this thesis. Furthermore, AxC-slots can be used to transport one or several IQ-samples belonging to the same carrier. The AxC-slot size can differ, but is always an even number of bits (20, 24 or 30). Which sizes that are supported depends on the capability of the concerned node. Carrier branches can hold one or many AxC-slots, all of which must use the same AxC-slot size. To help distinguish between the AxC-slot size and the number of AxC-slots within a carrier branch, the AxC-slot size is referred to as the IQ-format of the AxC-slot throughout this thesis. Thus, if we for example say that a carrier branch uses 20 bits IQ-format, it means that each AxC-slot belonging to that carrier branch uses an AxC-slot size of 20 bits.

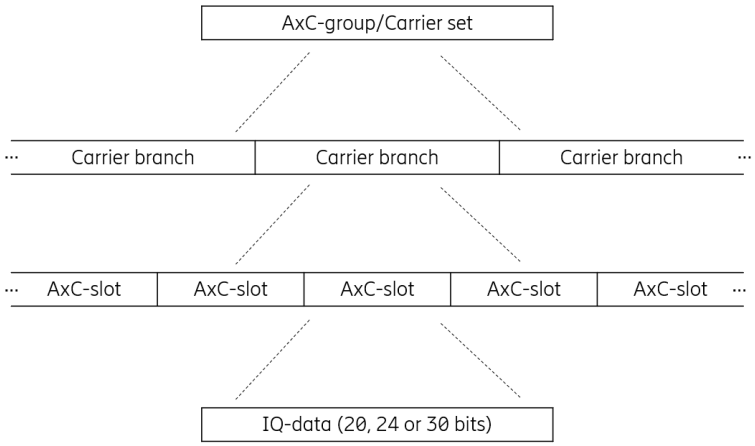


Figure 1.1.3: *The structure of an AxC-group.*

1.1.3 CPRI Frame Structure

CPRI uses a hierarchical frame structure with three kinds of frames. Every 10 ms, digital samples of a whole radio frame is transmitted using a CPRI frame. The CPRI frame is a collection of 150 hyperframes and constitutes the largest part of a CPRI link. Furthermore, one hyperframe consists of 256 basic frames.

In order to use different IQ-formats within a basic frame, one approach is to divide it into smaller partitions. For this, we introduce the IQ subframe concept, which is central for this thesis. Note that although CPRI is not vendor-specific, the implementations are. Thus, the concept with IQ subframes is not included in the official CPRI specification [7]. A single subframe can consist of 120 or 240 bits and are allocated by AxC-slots. Using subframes induces some constraints which are essential for this thesis. Among the most important are:

- A subframe does not support AxC-slots of mixed sizes.
- AxC-slots belonging to the same carrier branch can be allocated among consecutive subframes.
- AxC-slots using the same IQ-format, but originating from different carriers can share subframe.
- The available subframes within a basic frame are restricted by the line bit rate on the link.

Also note that allocating resources on one basic frame applies to all basic frames in the hyperframe as well as in the radio frame.

The CPRI frame structure and some of its constraints are illustrated in Fig. 1.1.4. Carriers 1 and 2 can share subframe 0 since both are using the 20-bit IQ-format. Furthermore, carrier 2 can be allocated among both subframe 0 and 1, as they are consecutive. At last, carrier 3 is allocated on a different subframe, since it uses another IQ-format.

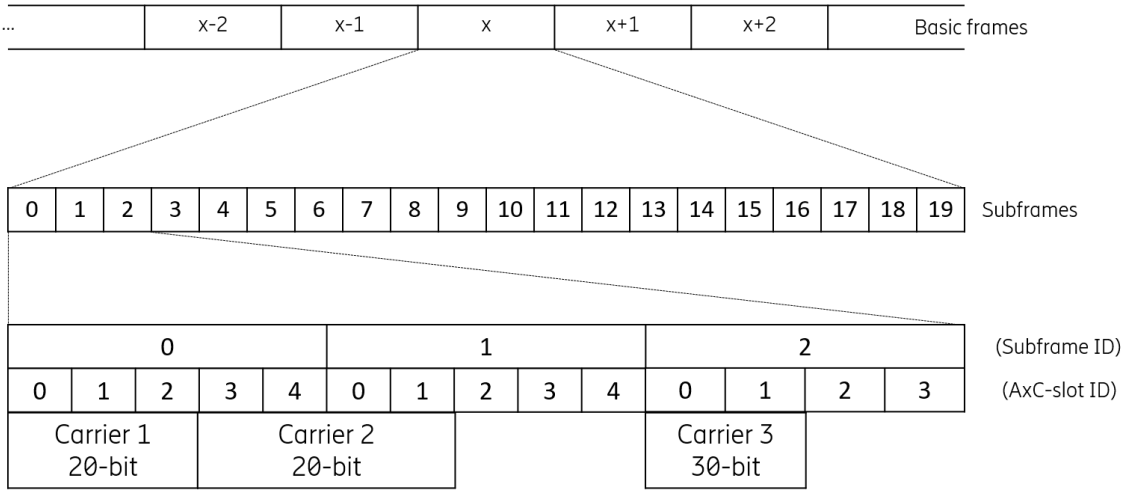


Figure 1.1.4: *Model of the CPRI frame structure using subframes, showing how three carriers has been allocated.*

1.1.4 CPRI Configurations

To get a view of possible RBS configurations, we go a bit further into the connection setups between REC and REs. More information on CPRI configurations can be found in the CPRI specifications [7]. The most trivial setup is shown in Figure 1.1.5. It consists of one REC and one or more REs, connected with one single point-to-point link, with line bit rate capacity l. Each link supports both uplink and downlink user data, which is illustrated with the double-headed arrow.

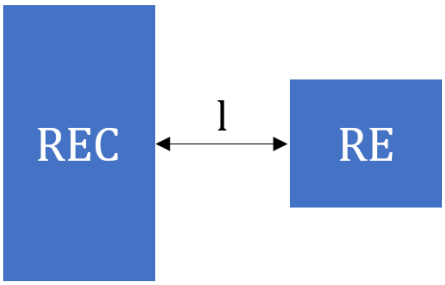


Figure 1.1.5: *A basic RBS configuration with a single point-to-point CPRI link.*

As mentioned in Section 1.1.1, RBS configurations can consist of multiple REC and RE units. Originating from the simple configuration in Figure 1.1.5, additional REs can be connected in series with RE_1, resulting in the chain topology seen in Figure 2.0.1. Here we denote the links with indices, where for example $l_{1,2}$ is the link between RE_1 and RE_2. The REC has index 0.

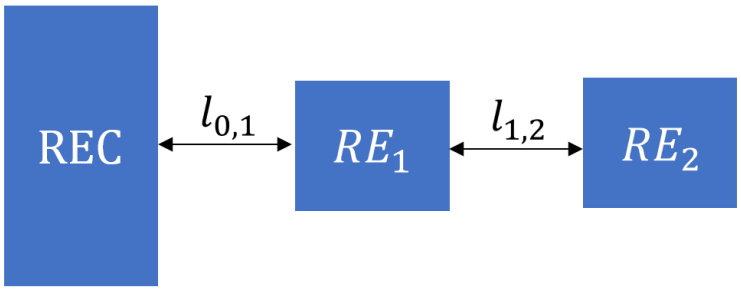


Figure 1.1.6: *An RBS configuration consisting of one REC and two REs connected with single point-to-point CPRI links using chain topology.*

In this case, carriers destined for RE_2 are forwarded via RE_1. Thus, the two REs RE_1 and RE_2 must share resources. Now assume that $l_{0,1} < l_{1,2}$, then some IQ-data forwarded from RE_1 to RE_2 may be out of range for $l_{1,2}$, resulting in loss of data. Since the available subframes within a basic frame are restricted by the line bit rate on the link, the allocation algorithm in this case should ensure that all carriers destined for RE_2 is allocated on subframes supported by $l_{1,2}$.

Normally, RBS configurations are designed to avoid bottle-necks. This means that for a chain topology, the line bit rates are non-increasing. That is, for a configuration of k REs, as shown in Figure 1.1.7, then $l_{i-1,i} \geq l_{i,i+1}$ for all $i = 1, ..., k-1$.

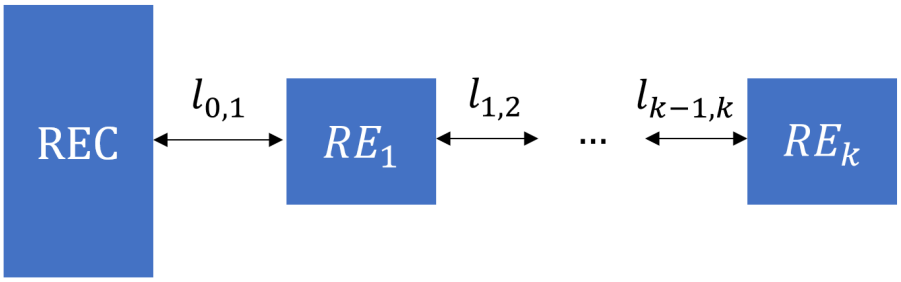


Figure 1.1.7: *A general RBS configuration using chain topology with single point-to-point links.*

1.2 Purpose

The purpose of this thesis work is to investigate how fragmentation related to the allocation of carrier branches onto CPRI basic frames can be reduced, in order to use radio interconnect resources more efficiently. In particular, the purpose is to evaluate potential advantages of considering multiple carrier requests before allocating, instead of allocating on a first-come-first-serve basis.

1.3 Problem Definition

This thesis concerns the problem of allocating a given set of carriers, i.e. carriers belonging to the same AxC-group, for a given RBS configuration. This includes the following two steps:

1. **IQ-format selection.** For each carrier in the set, a decision needs to be made on which IQ-format (AxC-slot size) to use.
2. **Subframe allocation.** A decision on which subframes to allocate for each carrier must be made.

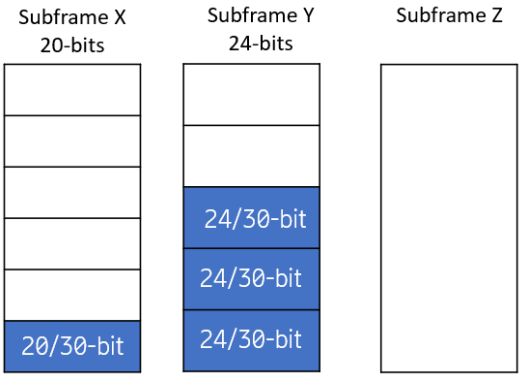
The pre-requisites for a proposed algorithm to perform these steps are:

- **Topology**. As described in Section 1.1.4, the complexity of RBS configurations is increasing. The topologies of which configurations to use needs to be defined.
- **Capabilities**. The supported IQ-formats for each unit in the topology must be known.
- **Link capacity**. The line bit rates of all links in the topology must be known.
- **Carrier capacity.** The required capacity for each carrier must be known.

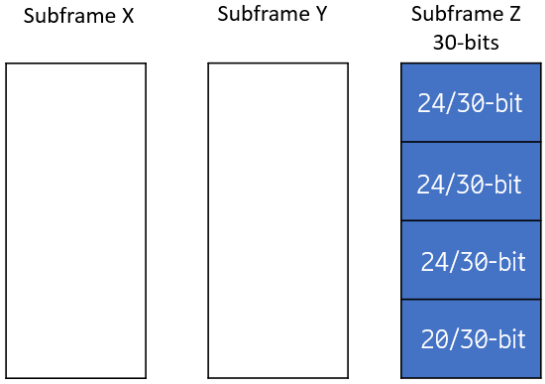
Furthermore, we differentiate between fragmentation due to non-optimal IQ-format selection described in Def. 1.3.1 and fragmentation due to the distribution of the carriers on the basic frame in Def. 1.3.2.

Definition 1.3.1. (Minimizing number of used subframes.) Given a set of carriers, select the IQ-format for each carrier such that the total number of used subframes is minimized.

In this problem, fragmentation is defined as using more subframes than necessary, due to non-optimal choice of IQ-format. This can be considered more of a theoretical problem where the selection of IQ-formats is evaluated without taking the actual allocation into consideration. In Fig. 1.3.1a and 1.3.1b, an example of fragmentation due to the selection of IQ-format is illustrated.



(a) *An example of the subframe allocation when the minimal format available is used.*



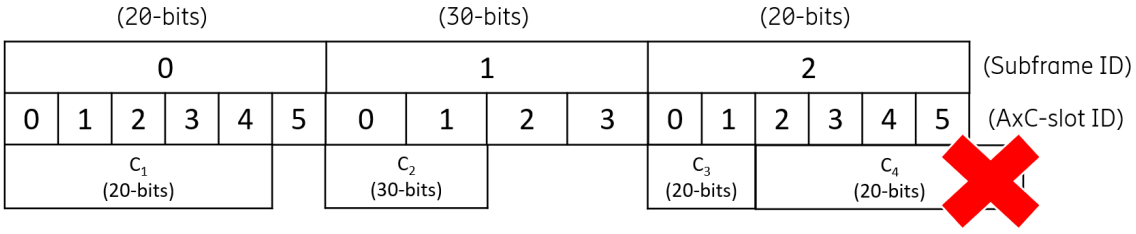
(b) *An example of how the number of used subframes can be reduced by using the same IQ-format.*

Figure 1.3.1: *An example of how IQ-format selection can reduce the number of used subframes.*

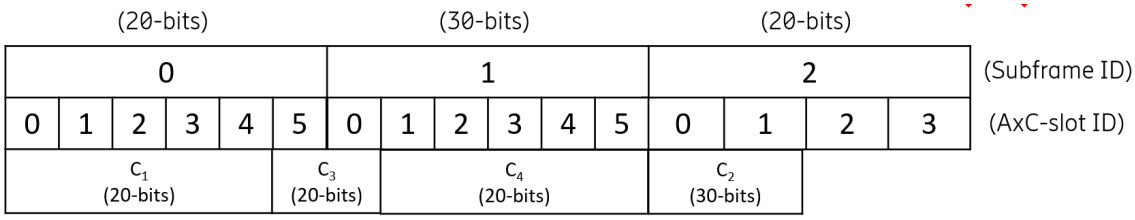
In Chapter 3, three solution alternatives for this problem is presented. For the next problem definition, fragmentation due to the actual allocation is considered.

Definition 1.3.2. (Optimizing disitribution of allocated carriers.) Given a basic frame and a set of carriers, distribute the carriers on the basic frame such that all carriers in the set can be allocated.

This is a practical problem, where we look at the ability of the algorithm to allocate a given set of carriers. Also here, fragmentation can occur if more subframes are allocated than needed. However, for this case it does not depend only on the IQ-selection, but also on how the carriers are distributed on the basic frame. In Fig. 1.3.2a and 1.3.2b, an example of how fragmentation can occur depending on in which order the carriers are allocated in is illustrated.



(a) *Due to fragmentation, c_4 cannot be allocated onto the basic frame.*



(b) *By rearranging the carrier distribution, fragmentation can be reduced allowing c_4 to be allocated.*

Figure 1.3.2: *Illustration of how fragmentation due to allocation order can prevent carriers to be allocated onto a basic frame.*

Fig. 1.3.2b shows how fragmentation can be avoided by using the right allocation strategy. Depending on the configuration, additional constraints must be considered for these algorithms, such as differing line bit rate within a configuration.

The two problems are evaluated using different parameters. For Problem 1.3.1, we look at:

- Used subframes. For a given carrier set, how many subframes are used.
- Used number of bits. To minimize fragmentation, using a non-minimal number of bits may be required.

For Problem 1.3.2, we look at:

- Successful allocations. For a given set of carriers, we are interested in whether all carriers can be allocated to the basic frame or not.
- Average number of allocated slots. The average number of AxC-slots that the algorithm is able to allocate before failing.

Furthermore, the constraints are:

- For each link, the number of used AxC-slots must not exceed the capacity of the link.
- For each subframe, only one IQ-format can be supported.

Regarding the allocation strategy, it may not be possible to determine an objectively optimal distribution. However, the solution must fulfill the criteria that every allocated subframe is supported by the line rate related to its AxC-slots.

In accordance with the purpose, potential advantages of going from a first-come-first-serve basis to considering all carriers at the same time is considered. In order to differentiate between these two cases, we define online and offline algorithms according to Def. 1.3.3 and 1.3.4.

Definition 1.3.3. (Online algorithm.) For a given set of carriers, the IQ-format selection

and allocation decision is taken for each carrier in the order that they appear in the set. Thus, the decision for each carrier is taken without considering any other carrier in the set.

Definition 1.3.4. (Offline algorithm.) For a given set of carriers, the IQ-format selection and allocation decision is done after considering all carriers in the set. Thus, the entire set needs to be known to the algorithm in order to take a decision for a single carrier.

1.4 Scope and Limitations

The optimum way of using resources on the transmission link can be defined in many different ways, such as maximizing user data throughput, minimizing the HW cost for a specific use case, minimizing the power consumption for a specific use case and minimizing required interconnect resources for a specific use case. The resource allocation considered in this thesis is mainly done during start-up or if a failure occurs in the system. Therefore, factors such as runtime are not highly prioritized for the resource allocation. Instead, the optimum way in this thesis is defined as minimizing the required interconnect resources. This means that the performance in terms of algorithm complexity is overlooked to a large extent. However, the area is rapidly changing, and future technology may for example be cloud-based. This could make the memory access less critical, and perhaps put higher demand on runtime.

Another reason for not measuring the performance in terms of runtime is that the algorithms are not implemented or tested on actual vendor specific systems. To avoid misleading results depending on the implementation, time benchmarking for the implemented algorithms in the simulation environment is not considered. Instead, the comparison between different solutions is limited to simulations using prototype algorithms. With this said, the complexity cannot be completely overlooked. The algorithms must be able to run in a realistic amount of time, both in order to be considered plausible, but also in order to be simulated.

Apart from allocating interconnect resources, the problem of resource allocation for large configurations are strongly connected to the allocation of radio and baseband processing resources. These areas are not considered in this thesis.

The problem of finding the shortest path for the carrier branch is outside the scope of this thesis. The path is assumed to given.

The available IQ-formats are limited to 20, 24 and 30 bits and the line bit rates to 2.5, 4.9, 9.8 and 10.1 Gbps.

As mentioned in Section 1.1.4, REs with switch functionalities are not considered. Thus, a master port on an RE cannot forward traffic to two different slave ports. Nor can two different master ports forward traffic to the same slave port. This put some limitations to which RBS configurations that need to be supported.

1.5 Ethical and Ecological Aspects

The environmental impact from the telecommunication industry is a growing concern. Although there are many positive aspects of better communication facilities, e.g. reducing unnecessary travelling, improvements are needed to be able to meet future demands

without compromising with the environment. Because of this, Ericsson works towards a number of climate-related objectives, including

- achieving 35 % energy saving in Ericsson Radio System by 2022 compared to 2016
- achieving a 5G product portfolio that is ten times more energy efficient (per transferred data) than 4G in 2022 compared to 2017
- reducing 35 % of CO_2e emissions from Ericsson's own activites by 2022 compared to 2016.

More information about this can be found in Ericsson's Sustainability and Corporate Responsibility Report from 2018 [5]. In this thesis, it is investigated how resources can be used more efficiently, which is in accordance with these sustainability goals.

1.6 Ongoing Research and the Future of CPRI

The requirements of the 5G NR radio access technology is a big challenge for fronthaul networks. With the transition from coaxial-based to fiber-optic transmission the fronthaul capacity was increased significantly. Recently, the expanding traffic has started to push the limits for fiber-optic transmission using serial interfaces such as CPRI [3]. Further reading on 5G requirements on fronthaul networks can be found in [10]. Although this thesis examines ways to improve CPRI implementations, it is worth mentioning other solutions that are being researched as well.

One way to increase the effective data rate in fronthaul networks is to compress the CPRI data. In [11], four times compression is achieved using a vector quantization algorithm. Other studies on CPRI compression includes [12] and [13].

The packet-oriented Ethernet-based CPRI (eCPRI) can be used to reduce the load on the fiber by splitting up the REC functionalities over the RBS, moving some processing to the RE. However, according to [14], eCPRI is not enough and there is need for even more drastic changes. In 2018, the IEEE 1914 standard for the next-generation fronthaul interface (NGFI) was presented, using Ethernet to achieve flexible and reliable fronthaul transmission, being able to fully support 5G technologies with virtualized RECs and massive MIMO. Unlike CPRI, which uses constant fiber bandwidth, Ethernet-based technologies can adapt to fluctuating mobile traffic load. Research on handling dynamic traffic in fronthaul networks can be found in [15][16][17][18]. With so much of the future technology focusing on eCPRI, it is fair to raise the question whether it is worth to continue developing the CPRI technology. However, even if CPRI would be outdated in the future, it is likely to take a long time for the industry to replace existing infrastructure, meaning that CPRI will play an important role for the network industry for many more years [19].

For the problems defined in this thesis, they are related to the vendor-specific implementation of CPRI. Altough CPRI is an open standard, it can be seen as a proprietary technology due to different implementations. Therefore, there is not much research done for the specific problems of this thesis in terms of CPRI. However, the problems are closely related to general concepts within combinatorial optimization, which can be examined to evaluate previous work.

1.7 Method

The investigation of suitable algorithms was based on a literature study, surveying solutions to similar allocation problems such as bin packing, knapsack and subset sum. Appropriate solutions was selected based on how well their attributes matched the constraints described in Section 1.3. Existing research on these problems were used in order to design the algorithms presented in this thesis.

In order to compare algorithms, a simulation environment was developed in Python 3.7. The purpose of this was to create a user-friendly test environment, allowing to generate random carrier requests and topologies within the framework, in order to simulate the performance of the algorithm alternatives.

1.8 Report Structure

Hereafter, the report will be presented as follows. First, the theoretical model that was used for CPRI is described in Chapter 2. In Chapter 3 the problem of fragmentation due to IQ-format selection is investigated and the developed algorithms are presented. The most important part of the report, Chapter 4, concerns the problem of allocating onto the basic frame, and presents the algorithms that was evaluated. Chapter 5 describes the simulation framework that was developed, and the test scenarios that was used for evaluating the algorithms. Then follows the results evaluating the algorithms in Chapter 6 and a discussion of the results in Chapter 7. Finally, the conclusions that were made from the thesis are presented.

2

Model of the CPRI Frame Structure

In order to implement the algorithms, a model covering the RBS configurations, carrier branches and the CPRI frame structure was developed. This chapter aims to describe this model, which is based on the CPRI background introduced in Section 1.1. First, we introduce the most essential definitions by letting

$\mathbf{C}$	be a set of carriers
c_i	be carrier i in $\mathbf{C}$
n	be the number of carriers in $\mathbf{C}$
w_i	be the number of AxC-slots in c_i
$\mathbf{S}$	be a set of subframes in a basic frame
s_j	be subframe j in $\mathbf{S}$
m	be the number of subframes in a basic frame
$\mathbf{T}$	be a set of IQ-format types, e.g. $\mathbf{T} = \{20, 24, 30\}$
$\sigma : \mathbf{S} \to \mathbf{T}$	be an indicator function, revealing the subframe type $t \in \mathbf{T}$ for a given subframe $s \in S$, i.e. $\sigma(s_j) = t$ iff subframe s_j is of type t
$\boldsymbol{\tau}_i$	be the supported formats for c_i, thus $\boldsymbol{\tau}_i \subseteq \mathbf{T}$
t_i	be the used format for c_i
λ_t	be the capacity of AxC-slots for a subframe of type t. Then, the capacity for s_j is given by $\lambda_{\sigma(j)}$.
$\mathbf{R}$	be the set of REs in a configuration
r_k	be RE k in $\mathbf{R}$
$\mathbf{L}$	be a set of line bit rates, e.g. $\mathbf{L} = \{1.2, 2.5, 4.9, 9.8\}$ [Gbps].
l_i	be the limiting line bit rate for carrier c_i in a configuration
$l_{i,j}$	be the line bit rate between r_i and r_j
$\phi : \mathbf{L} \to \mathbf{S}$	be a mapping function, revealing the maximum subframe index j for a given line bit rate according to Table 2.0.2.

Also, let

$$x_{i,t} = \begin{cases} 1 & \text{if } c_i \text{ is of type } t \in \mathbf{T} \\ 0 & \text{otherwise} \end{cases} \tag{2.0.1}$$

and

$$y_{i,j} = \begin{cases} 1 & \text{if carrier } c_i \text{ is in subframe } j \\ 0 & \text{otherwise} \end{cases} \tag{2.0.2}$$

$$q_{i,j} = \begin{cases} 1 & \text{if AxC-slot } i \text{ is used in subframe } j \\ 0 & \text{otherwise.} \end{cases} \tag{2.0.3}$$

The parameter values needed for the model are presented in Table 2.0.1, according to the limitations from Section 1.4.

Table 2.0.1: *Parameters values used for the CPRI model.*

Parameters	**Values**
IQ-formats	{20, 24, 30} [bits]
Basic frame sizes	{480, 960, 1920, 2400} [bits]
Subframe size	120 [bits]
Line bit rates	{2.4576, 4.9152, 9.8304, 10.1376} [Gbps]

Since the allocations on one basic frame apply to all basic frames in the hyperframe as well as in the radio frame, the entire allocation was modelled using only one basic frame. This basic frame was modelled as the set of subframes, **S**, that it holds. The number of subframes, m, in **S**, can be expressed as

$$m = \frac{\text{BASIC FRAME SIZE}}{\text{SUBFRAME SIZE}}. \tag{2.0.4}$$

As mentioned in Section 1.1.3, the basic frame size and the line bit rate are related. The exact relationships between them are presented in Table 2.0.2, along with the corresponding number of subframes according to Eq. 2.0.4.

Table 2.0.2: *The relations between line bit rate, basic frame size and number of subframes.*

Line bit rate [Gbps]	2.4576	4.9152	9.8304	10.1376
IQ-bits per basic frame	480	960	1920	2400
Number of subframes	4	8	16	20

A subframe, s_j, was modelled according to Table 2.0.3.

Table 2.0.3: *Model of a subframe, s_j.*

Subframe attributes	
Capacity	120 bits
Used format	t
Number of AxC-slots	λ_t
AxC-slots	$q_{0,j}, ..., q_{\lambda_t - 1,j}$

The number of AxC-slots in a subframe, λ, depends on which IQ-format the subframe uses. For a subframe using IQ-format t, we get

$$\lambda_t = \frac{\text{SUBFRAME SIZE}}{t}. \tag{2.0.5}$$

which yields the number of AxC-slots per subframe depending on the type according to Table 2.0.4.

Table 2.0.4: *The number of AxC-slots per subframe for each IQ-format type.*

t [bits]	λ_t [AxC-slots/subframe]
20	6
24	5
30	4

To determine the basic frame sizes, knowledge of all links in the RBS configuration is needed. An RBS configuration consisting of k REs was modelled as a set of REs, $\mathbf{R} = r_1, ..., r_K$, connected by links $l_{0,1}, ..., l_{K-2,K-1}$, where the 0:th index represents the REC. The specifications of the REC are not really needed for the model, and was assumed to be compatible with the rest of the configuration. Due to the bottle-neck constraint described in Section 1.1.4, only the limiting line bit rate for an RE, r_k, was considered of interest. The model of an RE is shown in Table 2.0.5.

Table 2.0.5: *Model of an RE, r_k.*

RE attributes	
Supported formats	$\boldsymbol{\tau}_{r_k}$
Limiting line bit rate	l_k

A typical problem instance is that a carrier set, $\mathbf{C} = \{c_1, ..., c_n\}$ is to be allocated onto a basic frame. A carrier was modelled by the attributes in Table 2.0.6.

Table 2.0.6: *Model of a carrier, c_i.*

Carrier attributes	
AxC quantity	w_i
Supported formats	$\boldsymbol{\tau}_i$
Used format	t_i
Destination RE	r_k

A carrier, c_i, with destination RE, r_k, must be allocated to a subframe, s_j, such that the subframe index j is supported by the limiting line bit rate, l_k, for r_k. This puts the constraint that

$$y_{i,j} = 0 \quad \text{for all } j > \phi(l_k). \tag{2.0.6}$$

The model is illustrated using a simple problem instance in Ex. 2.0.1.

Example 2.0.1. Assume an RBS configuration consisting of two REs with different line bit rates according to Fig. 2.0.1. According to the model, the set of REs is expressed as $\boldsymbol{R} = \{r_1, r_2\}$ with links $l_{0,1} = 4.9$ Gbps and $l_{1,2} = 2.5$ Gbps. According to Table 2.0.2, the maximal number of subframes is given by $m = \phi(l_{0,1}) + 1 = 8$. Thus, the basic frame is modelled as a set of subframes $\boldsymbol{S} = \{s_0, ..., s_7\}$. However, as can be seen in Fig. 2.0.1, the entire basic frame can only be used by $l_{0,1}$. For r_2, we get that $\phi(l_{1,2}) = 3$, which means that only subframe indexes $j = 0, ..., 3$ are available. The supported formats are $\boldsymbol{\tau}_{r_1} = \{30\}$ and $\boldsymbol{\tau}_{r_2} = \{20\}$ respectively.

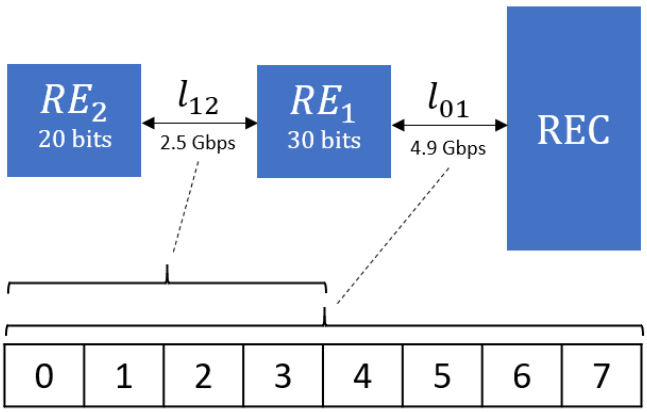


Figure 2.0.1: *Model of an RBS configuration, showing how different subframes are available depending on the line bit rates.*

Now assume that the following set of carriers requests to be allocated on the basic frame:

- c_1: 1 AxC-slot, destination: r_1
- c_2: 3 AxC-slot, destination: r_2

Since c_1 has r_1 as destination, its supported formats are $\boldsymbol{\tau}_1 = \boldsymbol{\tau}_{r_1} = \{30\}$ bits and can be allocated on any of the subframes. c_2 can only be allocated on the lowest four subframes. Assume that both carriers are allocated to their highest supported subframes, according to Fig. 2.0.2. c_1 is then allocated to s_7. Thus, the IQ-format of s_7 gets fixed to 30 bits, and we can define $\lambda_{\sigma(7)} \equiv \lambda_{30} = 4$ slots. Thus, the available AxC-slots in s_j are $q_{0,7}, ..., q_{3,7}$. In Fig. 2.0.2, c_1 is allocated to $q_{3,7}$ and c_2 to $q_{5,3}, ..., q_{3,3}$.

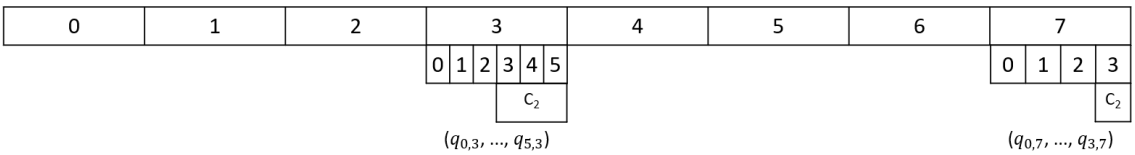


Figure 2.0.2: *Carrier allocations onto the basic frame from Fig. 2.0.1*

.

3

IQ-selection Algorithms for Minimizing the Number of Subframes

In this chapter, it is described how Problem 1.3.1, which was defined in Section 1.3 and handles the selection of IQ-formats for the carriers, was modelled. It is investigated how fragmentation due to non-optimal choices of IQ-formats can affect the number of needed subframes. The purpose of investigating this problem on its own was to see how close to the optimal solution we can get with different solutions. However, it is important to remember that in this chapter we are only concerned with the minimum number of subframes for each solution, disregarding fragmentation due to the distribution of the carriers onto the basic frame. In Chapter 4, the problem of the actual allocation is discussed. First, the problem and its constraints is discussed using the CPRI model presented in Chapter 2. Then, three proposed algorithms that was implemented are described.

3.1 Problem Definition and Constraints

As described in Chapter 2, the typical problem instance that we are dealing with is to allocate a set of n carriers, $\mathbf{C} = \{c_1, ..., c_n\}$ onto a basic frame, consisting of a set of m subframes, $\mathbf{S} = \{s_1, ..., s_m\}$. Typically, there are two choices to be made for each carrier; selecting IQ-format and deciding where on the basic frame it should be allocated. As we were only interested in the fragmentation due to non-optimal selection of IQ-formats in this chapter, the second step was disregarded. Then a solution, $\mathbf{x}_k$, could be defined solely by the selection of IQ-formats for each carrier as

$$\mathbf{x}_k = \begin{bmatrix} x_{1,t_1} & x_{2,t_2} & ... & x_{n,t_n}. \end{bmatrix} \tag{3.1.1}$$

The minimum number of needed subframes for a solution, $\mathbf{x}_k$, was achieved by assuming that all carriers using the same IQ-format can be allocated consecutively. Since a carrier can be shared among consecutive AxC-slots over different subframes, as long as the IQ-format is supported, we get that for n carriers using the same IQ-format, t, the maximum number of non-filled subframes is one, under the assumption that all carriers can be allocated consecutively, see Figure 3.1.1.

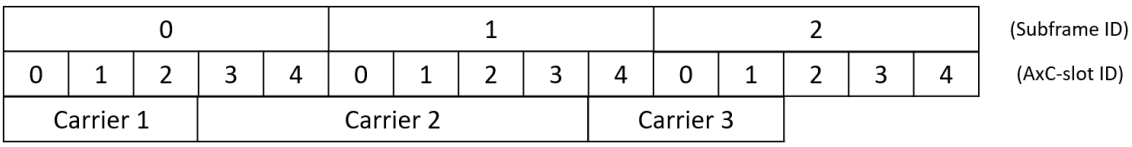


Figure 3.1.1: *Three carriers using IQ-format of 24 bits consecutively allocated onto AxC-slots, resulting in one non-filled subframe.*

If three different IQ-formats are used in the carrier set, the maximum number of non-filled subframes is instead three, assuming that carriers of the same IQ-format can be allocated consecutively. Using this assumption, the minimum number of needed subframes for a given solution, $\mathbf{x}$, to a problem instance, was calculated as

$$\sum_{t \in \mathbf{T}} \left\lceil \frac{\sum_i^n w_i \cdot x_{i,t}}{\lambda_t} \right\rceil \tag{3.1.2}$$

using the definitions from Chapter 2. Furthermore, the number of used bits for a given solution was calculated as

$$\sum_{t \in \mathbf{T}} \sum_{i}^{n} w_i \cdot x_{i,t} \cdot t. \tag{3.1.3}$$

Unlike the number of subframes, the number of bits for a solution is fixed, and does not rely on the assumption of consecutive allocation.

3.2 Problem Complexity

The problem of finding the optimal IQ-selection for a given carrier set under the assumptions described above, was formulated as finding the solution, $\mathbf{x}_k$, which minimizes Eq. 3.1.2. This has strong similarities with traditional combinatorial optimization problems such as the bin packing problem and the knapsack problem, where a given set of items with different attributes are packed into a set of bins or a 'knapsack', in an optimal way. Our problem can be defined as a special case of bin packing, with similarities to the bin packing with conflicts [20], multiple choices [21] or bin-dependent items [22], where different bin types are introduced and constraints are set on which items that can use which bin type. These combinatorial optimization problems are classed as NP-complete , which basically means that there is no known way of finding the optimal solution in polynomial time [23]. In our case this mean that the complexity of finding the optimal solution for a given carrier set is exponentially increasing with the number of possible solutions. By looking at the number of carriers supporting different number of IQ-formats, we got $1^{n_1} \cdot 2^{n_2} \cdot 3^{n_3}$ number of possible solutions for a problem instance, where n_i is the number of carriers supporting i different IQ-formats.

The most intuitive way of finding the optimal solution for NP-complete problems is naive brute-force, where all possible solutions are compared. Although there are techniques

for reducing the search space, such as branch-and-bound and cutting-plane, the worst-case complexity can still be equivalent to simple brute-force. The complexity of NP-complete problems has led to many alternative solving techniques, which allows to relax the optimality condition. Some example techniques used by algorithms are heuristics, randomization, parametrixation and approximation. Although the focus in this report was not on the complexity, the fact that there is a big risk that finding the optimal solution is not realistic for large carrier sets could not be ignored. Although it is not proven that our problem is in fact NP-complete, it was decided reasonable to focus on approximate algorithms. However, the optimal solution was still considered to be of interest in order to measure the quality of different algorithm alternatives.

3.3 Proposed Algorithms

In total, three algorithm for finding solutions to the IQ-selection problem was designed and implemented. The first one was the optimal alternative, while the latter two were approximative and does not guarantee optimality.

3.3.1 Optimal Algorithm (OPT)

As described in Section 3.2, the complexity of finding the optimal solution in terms of the minimum number of subframes for a set of carriers is exponentially increasing with the number of possible solutions. The optimal algorithm described in Alg. 1 was designed to guarantee that there is no other solution which can yield a lower minimum number of used subframes according to Eq. 3.1.2.

Algorithm 1 Optimal algorithm

```
MIN SUBFRAMES ← ∞
MIN BITS ← ∞
Let 𝒳 be the set of all solutions.
for each solution x_k ⊆ 𝒳 do
  calculate USED BITS and USED SUBFRAMES for x_k according to Eq. 3.1.3 and 3.1.2.
  if USED SUBFRAMES < MIN SUBFRAMES then
    BEST SOLUTION ← x_k
  else if USED SUBFRAMES = MIN SUBFRAMES and USED BITS < MIN BITS then
    BEST SOLUTION ← x_k
  end if
end for
return BEST SOLUTION
```

The MIN SUBFRAMES and MIN BITS variables keep track of the corresponding values for the best solution found so far. Going through all possible solutions, i.e. all combinations of IQ-formats for all carriers in the set, BEST SOLUTION is updated whenever a solution is found for which the minimum number of subframes is lower than the current value of MIN SUBFRAMES. For the case of equal subframes, the solution using the lowest number of bits is prioritized.

3.3.2 Minimal Format Algorithm (MF)

In many real-life applications, it is not vital to find the optimal solution every time. Therefore, approximation algorithms are common. The first approximate algorithm that was implemented was the minimal format algorithm (MF), described in Alg. 2. This is the simplest among the algorithms, for which the IQ-format selection of each carrier can be done independently from the rest of the set. This allows for online implementations, where the entire carrier set does not need to be known in order to do the selection for one carrier.

Algorithm 2 Minimal format (MF) heuristic

```
// x_k is the solution set to be returned, defined in Eq. 3.1.1.
x_k ← ∅
for each c_i ∈ C do
   // Assign c_i the minimal of its supported formats and add to x_k
   t_i ← min(τ_i)
   x_{i,t} ← 1
   Add x_{i,t} to x_k
end for
return x_k
```

The idea of this algorithm is that each carrier uses the minimal IQ-format that it supports. More formally, for a carrier, c_i, supporting IQ-formats $\boldsymbol{\tau}_i$, the format t_i is selected as $t_i = min(\boldsymbol{\tau}_i)$. The MF solution, $\boldsymbol{x}_{MF}$, for a set of n carriers, is expressed as

$$\boldsymbol{x}_{MF} = \{x_{1,min(\boldsymbol{\tau}_1)}, ..., x_{n,min(\boldsymbol{\tau}_n)}\}. \tag{3.3.1}$$

This algorithm can also be interpreted as minimizing the number of used bits according to Eq. 3.1.3. However, minimizing the number of bits does not guarantee optimality in terms of used subframes.

3.3.3 Favourable Changes Algorithm (FC)

The third IQ-selection algorithm that was designed and implemented was the favourable changes algorithm (FC), described in Alg. 3. When not taking the actual allocation of subframes into consideration, it was assumed that all carrier branches using the same IQ-format can be allocated continuously, and therefore also overlap among different subframes. Thus, there can never be more than one subframe per format which is not filled, i.e. no fragmentation on the link. So instead of looking at all possible solutions, FC is limited to moving around "excessive" AxC-slots to other formats.

Algorithm 3 Favourable changes (FC) heuristic

Run MF algorithm.
for each possible move (FROM FORMAT $\rightarrow$ TO FORMAT) **do**
  Calculate EXC(FROM FORMAT) and FREE(TO FORMAT) according to Eq. 3.3.2 and 3.3.3.
  if not EXC(FROM FORMAT) $\leq$ FREE(TO FORMAT) **then**
    continue
  end if
  // Create a set of candidate carriers which fulfills the favourable criterias.
  $\boldsymbol{C}_{candidate} \leftarrow \{c_i \in \boldsymbol{C}$ such that FROM FORMAT $\in \boldsymbol{\tau}_i \wedge$ TO FORMAT $\in \boldsymbol{\tau}_i \wedge \lambda_i \leq$ FREE(TO FORMAT)$\}$
  Try to find a subset $C_k \subseteq \boldsymbol{C}_{candidate}$ such that EXC(TO FORMAT) $\leq \lambda_{C_k} \leq$ FREE(TO FORMAT) by solving the subset sum problem.
  if $C_k \exists$ **then**
    Set TO FORMAT as used format for all $c_i \in C_k$
  end if
end for

FC was designed to utilize the principle of MF that it is beneficial to use the minimal format to as large extent as possible. However, FC enables to violate this principle in a small scale. The first step of FC is to run the MF algorithm and then originate from that solution. At this point, the only possible changes of IQ-format are upwards, i.e. 20 $\rightarrow$ 24, 24 $\rightarrow$ 30 and 20 $\rightarrow$ 30. The idea is to check for IQ-format changes that can be favourable in terms of the number of used subframes. This is explained in Ex. 3.3.1.

Example 3.3.1. Assume that there are ten carriers on the link:

- c_1: 1 AxC-slots, 30-bit IQ-format
- c_2: 2 AxC-slots, 20/24/30-bit IQ-format
- c_3: 5 AxC-slots, 20/24/30-bit IQ-format
- c_4: 3 AxC-slots, 24/30-bit IQ-format
- c_5: 4 AxC-slots, 20-bit IQ-format
- c_6: 1 AxC-slots, 24/30-bit IQ-format
- c_7: 1 AxC-slots, 24/30-bit IQ-format
- c_8: 3 AxC-slots, 20-bit IQ-format
- c_9: 2 AxC-slots, 20-bit IQ-format
- c_{10}: 2 AxC-slots, 24/30-bit IQ-format

Running the MF algorithm results in the distribution shown in Table 3.3.1, according to Eq. 3.1.2 and 3.1.3.

Table 3.3.1: *The distribution for the set of carriers using minimal format.*

	20-bits	**24-bits**	**30-bits**
Used subframes	3	2	1
Allocated AxC-slots	16/18	7/10	1/4

In the distribution in Table 3.3.1, there are three "opened" subframes, i.e. not fully used. The next step is to check if it is possible to fit any excessive AxC-slots into another opened subframe, which basically would mean that it is possible to close one subframe without opening a new one. In order to preserve the objective of minimizing the number of used bits, FC looks for carriers that can be moved from 20 $\rightarrow$ 24 bits format. The first step is to check if it is theoretically possible to fit the excessive AxC-slots in the 20-subframe to the free AxC-slots in the 24-subframe. The number of excessive slots for a format, t, is calculated as

$$\text{EXCESSIVE SLOTS} = \sum_{i}^{n}(w_i \cdot x_{i,t}) \bmod \lambda_t \tag{3.3.2}$$

and the free slots as

$$\text{FREE SLOTS} = \lambda_t - \text{EXCESSIVE SLOTS}. \tag{3.3.3}$$

From Table 3.3.1 and Eq. 3.3.2, the number of excessive 20-bit AxC-slots are $16 \bmod 6 = 4$ and the number of free 24-bit AxC-slots are $5 - (7 \bmod 5) = 3$ according to Eq. 3.3.3. In order for a move to be favourable, it most hold that

$$\text{EXCESSIVE SLOTS} \leq \text{FREE SLOTS} \tag{3.3.4}$$

which is not true in this case. Next step is to check for 24 $\rightarrow$ 30 moves. In this case, the number of excessive 24-bits AxC-slots are $7 \bmod 5 = 2$ and the free 30-bits AxC-slots are $4 - (1 \bmod 4) = 3$. Since $2 < 3$, a favourable move in this direction is theoretically possible according to Eq. 3.3.4. To know if it is possible in practice, the carriers must be examined, since the AxC-slots belonging to the same carrier cannot use different IQ-formats. To see if there are any carriers that can be moved from 24 to 30, a set $\mathbf{C}_{candidate} = \{c_2, c_4, c_6, c_7, c_{10}\}$ of candidate carriers is created. Note that c_3 is not included, as it holds five AxC-slots, which exceeds the number of free slots. The next step is to find a subset $\mathbf{C}_k \subseteq \mathbf{C}_{candidate}$ such that

$$\text{EXCESSIVE SLOTS} \leq W_{\mathbf{C}_k} \leq \text{FREE SLOTS} \tag{3.3.5}$$

where $W_{\mathbf{C}_k}$ is the total number of AxC-slots in $\mathbf{C}_k$. This is the subset sum problem, which is a classical decision problem and is not further explained in this report. To find $\mathbf{C}_k$, $\mathbf{C}_{candidate}$ is sorted in terms of AxC-slots in ascending order. If a subset is found that fulfills Eq. 3.3.5, the IQ-format of the carriers in $\mathbf{C}_k$ is changed. In this case it exists a subset $\mathbf{C}_k = \{c_6, c_7\}$ such that $W_{\mathbf{C}_k} = 2$. Thus, the formats of c_6 and c_7 are changed to 30-bits and we get the distribution in Table 3.3.2.

Table 3.3.2: *The distribution for set* $\mathbf{S}$ *after moving carriers* c_6 *and* c_7 *to 30-bits.*

	20-bits	**24-bits**	**30-bits**
Used subframes	3	1	1
Allocated AxC-slots	16/18	5/5	3/4

Now, only five subframes are used in total. The last step is to check for changes from $20 \rightarrow 30$ bits. Since $16 \bmod 6 = 4 > 4 - (3 \bmod 4) = 1$, this is not possible according to Eq. 3.3.4.

In Fig. 3.3.1, an example of an FC move is illustrated.

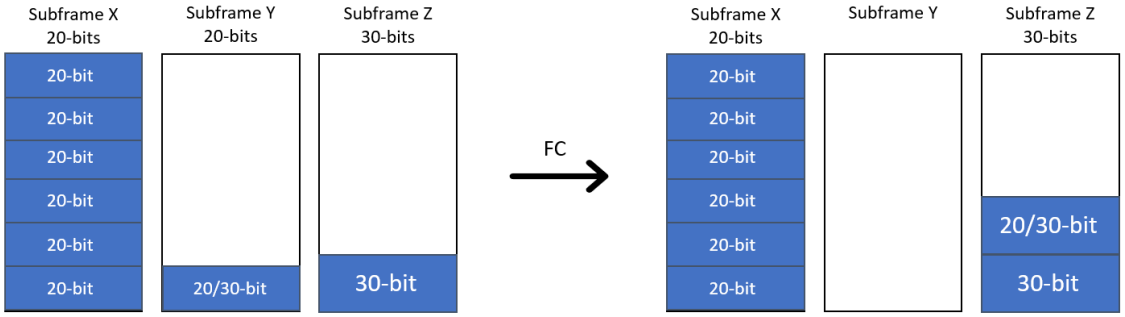


Figure 3.3.1: *An example where exc(20)* $= 1 \leq$ *free(30)* $= 3$ *and a carrier,* c_i*, supporting both IQ-formats and for which* $1 \leq w_i = 1 \leq 3$ *is found and moved.*

4

Allocation Algorithms

In Chapter 3 we looked at algorithms for IQ-selection in order to see how fragmentation due to IQ-selection can be avoided. An important assumption was that carriers using the same IQ-format could be allocated consecutively. In a real implementation, this is not always true. Depending on the allocation strategy, fragmentation can also occur due to the distribution of the carriers on the basic frame. This can happen for example if carriers are allocated in the wrong order, or if there are different line bit rates in the cascade so that carriers cannot be allocated freely among the subframes. In this chapter, the problem of fragmentation due to the distribution of carriers on the basic frame is investigated. Five different algorithms for the allocation that was implemented are described.

4.1 Problem Definition and Constraints

As before, we have the typical problem of allocating a set of n carriers, $\boldsymbol{C} = \{c_1, ..., c_n\}$, onto a set of m subframes $\boldsymbol{S} = \{s_1, ..., s_m\}$, which is performed in two steps; IQ-selection and AxC-slot allocation. A solution for the allocation problem was defined by both the selected IQ-format for each carrier and the coordinates on the basic frame that each carrier is allocated on.

In Chapter 2, the CPRI constraints due to link capacities in the cascade was defined. From this, some further constraints for a given carrier set, $\boldsymbol{C}$, was derived. A theoretical upper bound on the carriers using a link was set by seeing that the number of IQ-bits belonging to carriers with a certain line bit rate, l, cannot exceed the capacity of the supported subframes for l. Then for a subset $\mathbf{C}_k \subseteq \mathbf{C}$ such that $l_i \leq$ some l_k for all $c_i \in \mathbf{C}_k$, it must hold that

$$\sum_{i \in \mathbf{C}_k} w_i \cdot t_i \leq \phi(l_k) \cdot \text{SUBFRAME SIZE}. \tag{4.1.1}$$

For configurations with differing line bit rates, conflicts may occur depending on the objective. This is illustrated in Ex. 4.1.1.

Example 4.1.1. Assume that the following two carriers requests to be allocated:

- c_1: 3 AxC-slots, 20 bit IQ-format, 2.5 Gbps
- c_2: 3 AxC-slots, 20 bit IQ-format, 9.8 Gbps

The maximum subframe index is $\phi(2.5) = 3$ for c_1 and $\phi(9.8) = 15$ for c_2. In terms of minimizing the number of used subframes, the obvious choice is to put both carriers in

the same subframe s_j, where $j \leq 3$, as shown in Fig. 4.1.1. Then only one subframe is needed for the carrier set. This is the way Eq. 3.1.2 calculates the subframe, and is the assumption in Chapter 3. The problem with this is that the full capacity of l_2 is not utilized, which may cause problems for larger carrier sets.

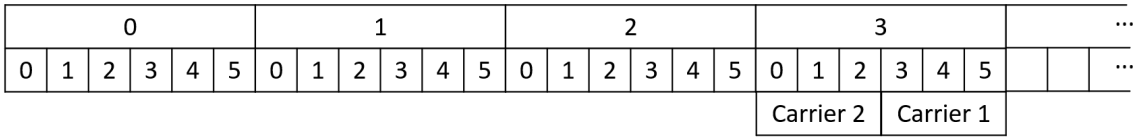


Figure 4.1.1: *In order to minimize the number of subframes, c_1 and c_2 is allocated onto the same subframe, although they are using different line bit rates.*

It is important to remember that the reason for reducing the number of used subframes is to increase the chances of successfully allocate all carriers in the set. If the capacities of different links are not taken into consideration when allocating, there is a risk that lower subframes are filled quickly, ending up in a scenario where some carriers cannot be allocated, although there is a lot of free space on the higher subframe indices. Assume for example that the carrier set is increased with two additional carriers, resulting in:

- c_1: 3 AxC-slots, 20 bit IQ-format, 2.5 Gbps
- c_2: 3 AxC-slots, 20 bit IQ-format, 9.8 Gbps
- c_3: 15 AxC-slots, 20 bit IQ-format, 2.5 Gbps
- c_4: 8 AxC-slots, 20 bit IQ-format, 2.5 Gbps.

Then there are in total 29 AxC-slots using 20 bit IQ-format in the set. According to Eq. 3.1.2, the minimum number of needed subframes is $\lceil 29/6 \rceil = 5$. Assume that the carriers are allocated in the order that they arrive, and that we want to minimize the number of used subframes. Then, we want to allocate the carriers consecutively, starting from the highest possible subframe ID. For the first two carriers, the highest common supported subframe ID is s_3, which both carriers will be allocated in. Moving on to c_3, it only supports $j \leq 3$, where s_3 is already allocated. Thus, it will be allocated among s_2, s_1 and s_0, according to Fig. 4.1.2.

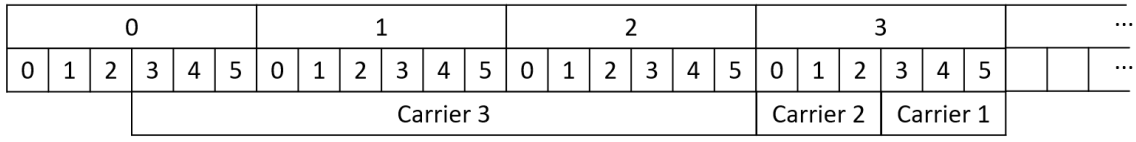


Figure 4.1.2: *Three carriers allocated among the four lowest subframe IDs, leaving only three free AxC-slots.*

Now, there is only three AxC-slots left among the lower subframes. Thus, c_4 cannot be allocated to the basic frame. Remember though, that there are in total 16 subframes in the basic frame, due to the line bit rate of 9.8 Gbps for c_2, which means in total $(16 - 4) \cdot 6 = 72$ additional free AxC-slots. If c_2 would have been allocated among them instead, there would be enough space for c_4 in the lower region. This example illustrates how important the allocation strategy is, and that we can not only rely on minimizing the number of subframes as was the case for IQ-selection.

In addition, we also have the fragmentation due to different IQ-formats. This means that there are two main issues that the allocation algorithms needs to consider; fragmentation due to mixing IQ-formats and resource problems due to line bit rate preference.

4.2 Allocation Algorithms

As discussed in Chapter 3, the distribution of carriers on the basic frame can be described as a bin packing problem with constraints. In terms of optimal solutions for the allocation algorithms, it can be defined in different ways depending on the objective. If for example an optimized solution is defined as maximizing the amount of free capacity in terms of bits on the basic frame, that would contradict the problem of minimizing the subframes described in Chapter 3. For this problem, it was considered interesting whether the allocation algorithm manages to allocate all carriers in the set or not. Finding an optimal solution to this using a brute-force approach like in Section 3.3.1 would mean considering every possible solution, i.e. every possible combination of IQ-format and allocation slot for each carrier. With a maximal basic frame size of 2400 bits and minimal slot size of 20 bits, there are potentially 120 slot positions on a basic frame. Searching through all possible solutions was not considered realistic.

In this section, three online algorithms are presented and then two offline alternatives. Note that 'online' in this case only considers the allocation part of the algorithm. For the entire chain to be online, the IQ-selection algorithm must be online as well.

4.2.1 First-Fit Algorithms

The first two algorithms are based on the first-fit (FF) algorithm, which is a well-known bin packing approximation algorithm [24]. The idea is that for each carrier in the set, the algorithm looks for the first available sequence of AxC-slots in the basic frame. This can be done in different ways by choosing the starting coordinates $q_{i,j}$, where j is the subframe index and i is the AxC-slot index for subframe j, as defined in Chapter 2. The starting coordinates is basically where in the basic frame the algorithm should start looking from. The two ways of doing this that was implemented is to start allocating either from high or low subframe indexes, as in Strategy 4.2.1 and 4.2.1.

Strategy 4.2.1. (High first-fit strategy (HFFS)) For a carrier c_i, start FF from subframe index given by $\phi(l_i)$, i.e. the highest supported subframe index for the limiting line bit rate of the destination RE for c_i.

Strategy 4.2.2. (Low first-fit strategy (LFFS).) For a carrier c_i, start FF from lowest subframe index on the basic frame.

As for the entire carrier set, this was implemented as described in Alg. 4 and Alg. 5 for HFF and LFF respectively.

Algorithm 4 HFF algorithm

Decide IQ-selection algorithm
for each $c_i \in \boldsymbol{C}$ **do**
  // Set the highest available subframe as start index.
  $j \leftarrow \phi(l_i)$
  START COORDINATE $\leftarrow q_{\lambda_{t_i},j}$
  Allocate c_i using HFF from START COORDINATE.
end for

Algorithm 5 LFF algorithm

Decide IQ-selection algorithm
for each $c_i \in \boldsymbol{C}$ **do**
  // Set the lowest subframe as start index.
  START COORDINATE $\leftarrow q_{0,0}$
  Allocate c_i using LFF from START COORDINATE.
end for

As defined in Chapter 2, q refers to a specific AxC-slot in a subframe. λ_{t_i} is the highest available AxC-slot index for a subframe using IQ-format t_i, i.e. the selected IQ-format for the carrier c_i. On their own, these algorithms may seem trivial. However, by processing the carrier set and choosing IQ-selection they could easily be modified, and provided a basis for all the later allocation algorithms.

4.2.2 Combined First-Fit Algorithm (CFF)

Combined first-fit (CFF) is an online algorithm, and can therefore take decisions for each carrier, without considering future carriers. As the name implies, it is very closely related to HFF and CFF. However, it also takes the complexity of the RBS configuration into consideration, as well as past carrier allocations.

The main idea for CFF was to avoid fragmentation due to mixing IQ-formats, trying to achieve consecutive allocation of carriers with the same IQ-format, without having to sort them first. It tries to make use of as high subframe indices as possible, in order to avoid the problems due to different line bit rates that we saw in Ex. 4.1.1. CFF keeps track of the current state of the allocation, and then decides the allocation strategy from that. The key variables that defines the state is the number of unique line bit rates on the cascade and the number of used IQ-formats in the carrier set.

The starting allocation strategy is always HFF, so that first carrier is always be allocated on the highest possible subframe ID and AxC-slot ID. In terms of IQ-format selection, the starting algorithm is to use the minimal format, MF (see Section 3.3.2). Depending on the configuration, the allocation strategy and IQ-format selection may vary as more carriers arrive. One of the pre-requisites for the algorithm is the RBS configuration. The number of unique line bit rates in the cascade is known from the start and two states are defined depending on this information. If the number of line bit rates is three or more, then the HFF and MF is used for the entire carrier set, regardless of the other carriers. Otherwise, the algorithm may change depending on the total number of IQ-formats used in the carrier set. This is not known at the start however. The structure of CFF is described in Alg. 6.

Algorithm 6 CFF algorithm

if three or more different line bit rates **then**
  Use HFF and MF for all c_i
  return
end if
// The minimal format of the first carrier, c_1, sets the high format, th.
$th \leftarrow min(\boldsymbol{\tau}_1)$
Initialize starting state to 1, i.e. one used IQ-format.
for $c_i \in \boldsymbol{C}$ **do**
  State 1: only one used IQ-format.
    if $t_i = th$ **then**
      Allocate c_i using HFF.
    else
      // Set the low format tl to t_i.
      $tl \leftarrow t_i$
      Set START IDX to the lowest subframe ID that is not supported by any lower line bit rates in the configuration.
      Try allocating c_i using LFF first from START IDX and then from 0.
      Update state to 2.
    end if
  State 2: two IQ-formats used.
    if $t_i = th \vee t_i = tl$ **then**
      Allocate using HFF or LFF respectively.
    else
      if c_i supports any other IQ-format **then**
        Use that and allocate with HFF or LFF depending on the format.
      else
        Reallocate all carriers using HFF.
        Update state to 3.
      end if
    end if
  State 3: three IQ-formats used.
    Use HFF.
end for

CFF is illustrated in Ex. 4.2.1 below.

Example 4.2.1. Assume the RBS configuration:

- r_1: 30 bits, 4.9 Gbps
- r_2: 20/30 bits, 4.9 Gbps
- r_3: 30 bits, 2.5 Gbps
- r_4: 24/30 bits, 2.5 Gbps
- r_5: 20 bits, 2.5 Gbps

and five carriers on the link:

- c_1: 4 AxC-slots, 30 bits, 2.5 Gbps (r_3)
- c_2: 2 AxC-slots, 24/30 bits, 2.5 Gbps (r_4)

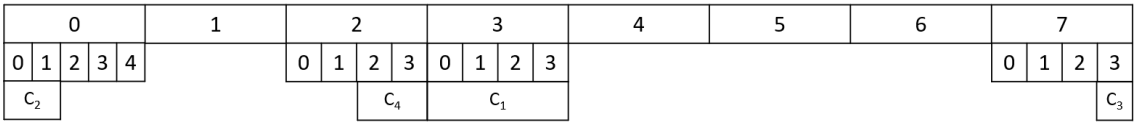
– c_3: 1 AxC-slot, 30 bits, 4.9 Gbps (r_1)

– c_4: 2 AxC-slots, 20/30 bits, 4.9 Gbps (r_2)

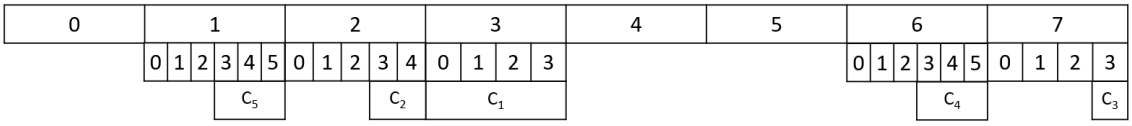
– c_5: 3 AxC-slots, 20 bits, 2.5 Gbps (r_5)

There are two different line bit rates in the cascade, so the strategy is not fixed from start. We look at the steps for allocating each carrier.

1. In the initial state, with two unique line bit rates, the allocation strategy is HFF and IQ-selection algortihm is MF. The highest available subframe ID for c_1 is given by $\phi(2.5) = 3$, so c_1 is allocated with HF, starting from s_3, with IQ-format 30 bits.
2. After allocating c_1, there is currently one used format in the carrier set. Using MF, c_2 will use 24 bits IQ-format. 24 bits would be the second IQ-format in the set. To avoid fragmentation, carriers using 24 bits on the 2.5 Gbps link is now allocated using LFF, starting from s_0.
3. We are now in the state of two different IQ-formats in the set, using HFF for 30 bit carriers and LFF for 24 bit carriers. For c_3, only 30 bits is available, and it is allocated using HFF. The maximum available subframe is $\phi(4.9) = 7$ for c_3, which is allocated with HFF, starting from s_7.
4. For c_4 we can see that the MF algorithm will choose 20 bits as IQ-format. However, that would increase the number of IQ-formats in the set to three. To maintain the allocation from two different directions, the MF IQ-selection is violated, and 30 bits will be used as IQ-format. c_4 is therefore allocated using HFF from s_7. The basic frame after four carriers have been allocated is shown in Fig. 4.2.1a.
5. For the fifth carrier, there is no longer a choice to maintain the state of only two different IQ-formats in the set. In the set of three different IQ-formats in the set, all carriers will be allocated using HFF, as in the case of three different line bit rates in the configuration. This step also involves a reallocation of all the previously allocated carriers, using MF as IQ-selection and HFF as allocations strategy. The resulting allocation is shown in Fig. 4.2.1b.



(a) *The distribution of carriers on the basic frame after step four in Ex. 4.2.1.*



(b) *The resulting distribution of carriers after step five in Ex. 4.2.1*

Figure 4.2.1: *An example of how CFF can be forced to reallocate using HFF in the case of three different IQ-formats.*

4.2.3 Sorting Algorithm (SA)

The next algorithm that was implemented for allocation is shown in Alg. 7. It originates from the idea that fragmentation can be avoided by allocating carriers of the same IQ-formats consecutively, as discussed in Chapter 3. To achieve this, the carriers are sorted

by IQ-format before being allocated. In terms of IQ-selection, it can easily be varied among the alternatives in Chapter 3. SA can be modified in many different ways, and is not dependent on the strategy or IQ-selection algorithm. It should be viewed as a modification to other algorithms, rather than a stand-alone algorithm.

Algorithm 7 Sorting algorithm (SA)

Decide IQ-format for each carrier in the carrier set using any IQ-selection algorithm.
Sort the carriers by IQ-format.
Allocate using any strategy.

4.2.4 Region-based Allocation Algorithm (RB)

In order to avoid fragmentation more efficiently without using unnecessary low subframe indexes, the RB algorithm in Alg. 8 was developed. It can be seen as a further development of the algorithms previously presented.

Algorithm 8 RB algorithm

// Divide the carriers into subsets depending on their line bit rate.
Let $\mathbf{C}_k \subseteq \mathbf{C}$ such that $\forall\, c_i \in \mathbf{C}_k$ it holds that $l_{c_i} = l_k \,\forall\, l_k \in \mathbf{L}$
for each region pair **do**
  For each IQ-format $f \in \mathbf{F}$, calculate EXC SLOTS and FREE SLOTS for each region.
  Find a subset of carriers from the higher region with total slots, w, such that EXC SLOTS $\leq w \leq$ FREE SLOTS.
  If subset found, move carriers to lower region.
end for
for each region pair **do**
  for $f \in \boldsymbol{F}$ **do**
    If non-optimal in terms of subframes, connect the pair.
    break
  end for
end for
Allocate using SA/LFF.

The idea is that all carriers are divided into different regions, which is basically the line bit rate of the RE that they belong to. For each region, an IQ-selection algorithm is run. Then, the theoretical number of subframes that would be needed if the carriers was to be allocated by SA/HFF, i.e. using the highest available subframe index in the region, is calculated. Next, for each IQ-format, it is decided whether the carriers uses more subframe than needed. This is defined as if there are more than one subframe using the IQ-format which is not filled, then there is fragmentation. If there is such a violation we look for favourable changes from one region to another. If it is possible to move carriers from a higher region to a lower, such that the total number of subframes used for the IQ-format is reduced, then the move is done. Note that this means that some carriers may be allocated to a lower region that needed. When this has been done for all IQ-formats, the carriers are sorted in order to create consecutive connection between regions. For an IQ-format pair in two nearby regions, where both have unfilled subframes, these are sorted to be consecutive. The RB allocation algorithm is illustrated in Example 4.2.2 below.

Example 4.2.2. Assume a configuration using chain topology with single point-to-point links with the following REs:

- r_1: 20-bit IQ-format
- r_2: 20-bit IQ-format.

The two links in the cascade are then:

- $l_1 = 2.5$ Gbps
- $l_2 = 4.9$ Gbps.

Also assume that a set, **C**, with carriers

- c_1: 2 AxC-slots, r_1
- c_2 5 AxC-slots, r_2
- c_3: 3 AxC-slots, r_2

requests to be allocated on the basic frame. There are two regions available, with maximum subframe indexes $\phi(l_1) = 3$ and $\phi(l_2) = 7$, in accordance with Fig. 4.2.2.

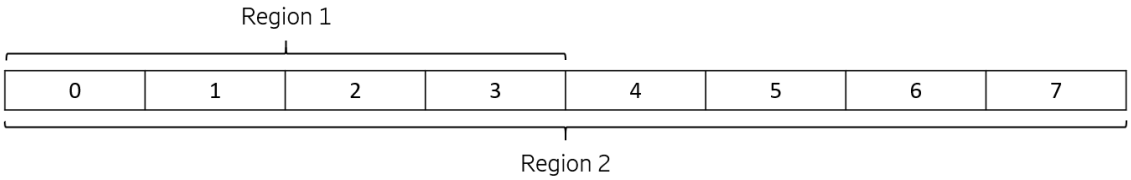


Figure 4.2.2: *Figure illustrating how the basic frame for an RBS configuration with line bit rates 2.5 and 4.9 Gbps is divided into two regions by the RB algorithm.*

Thus the regions are $j = 0, ..., 3$ for c_1 and $j = 0, ...7$ for c_2 and c_3. In Table 4.2.1 the number of subframes and AxC-slots for each region is shown.

Table 4.2.1: *The theoretical number of subframes and AxC-slots needed for allocation of the 20-bits carriers for each region.*

	Region 1	Region 2
Used subframes	1	2
AxC-slots	2/6	8/12

Without the line bit rate constraint, the number of subframes needed for the carriers would instead be

$$\left\lceil \frac{2+8}{6} \right\rceil = 2.$$

We have two non-filled subframes for 20-bit IQ-format, which means fragmentation. The next step is to relax the region constraint, and see if there are any plausible moves of carriers to a lower region. Moves to higher regions are not possible due to the line bit rate constraint. A plausible move is defined as a move which reduces the total number of used subframes. In the scenario in Table 4.2.1, there are two excessive slots in region

2 and four free slots in region 1. If c_2 was moved to region 1, only one subframe would be needed in region 2, but a new one would be opened in region 1, which is not desired. However, if c_3 is moved as in Fig. 4.2.3, it would fit in the opened subframe in region 1, and remove one subframe from region 2.

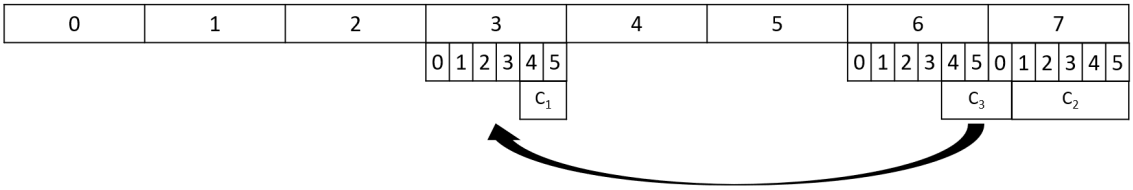


Figure 4.2.3: *A figure showing how c_3 can be moved to a lower region in order to reduce the number of subframes.*

The result after this move is shown in Table 4.2.2

Table 4.2.2: *The theoretical number of subframes and AxC-slots needed for allocation of the 20-bits carriers for each region after c_3 was moved to region 1.*

	Region 1	Region 2
Used subframes	1	1
AxC-slots	5/6	5/6

The total number of subframes needed for the allocation of **C** has been reduced from 3 to 2, at the cost of allocating 3 AxC-slots to a lower region than necessary.

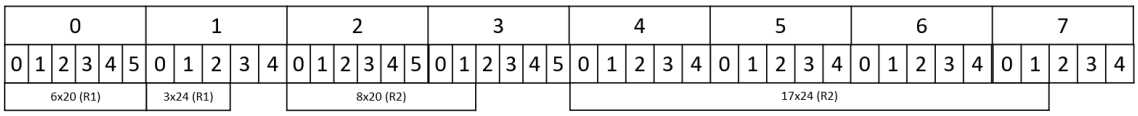
Before the allocation, the carriers are sorted by region and IQ-format. Then, in order to make use of potential connections between regions, we look for IQ-format in different regions which can be connected in the allocation phase. This sorting is illustrated in Ex. 4.2.3 below.

Example 4.2.3. Assume that we have the AxC-slots in Table 4.2.3 for each region and IQ-format available.

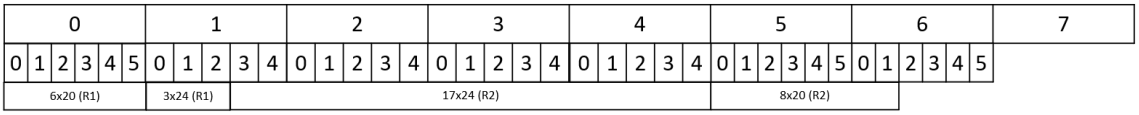
Table 4.2.3

	Region 1	Region 2
20-bit AxC-slots	6	8
24-bit AxC-slots	3	17

Then we sort and allocate using SA/LFF by region and IQ-format, such that 20 bit and 24 bit slots from region 1 is allocated first, and then 20 bit and 24 bit from region 2, in that order, according to Fig. 4.2.4a. Then seven subframes will be used in total. While if instead connecting the 24 bit IQ-formats, as in Fig. 4.2.4b, only six subframes are used.



(a) *Allocation without region connection.*



(b) *Allocation with region connection.*

Figure 4.2.4: *Two examples showing how RB avoids fragmentation by using region connection.*

This sorting is done by looking at the region and IQ-format pairs, one at a time. In this case, starting with 20-bit slots in region 1 and 2, we see that the minimum number of needed subframes according to Eq. 3.1.2 is $\lceil (6+8)/6 \rceil = 3$, which is the same as in Fig. 4.2.4a. Thus, there is no need to connect these. For the 24-bit pair however, the minimum number of subframes is $\lceil (3+17)/5 \rceil = 4$. However, in Fig. 4.2.4a, five subframes is used for these slots. Then, by sorting the slots in region 1 as 20, 24 and in region 2 as 24, 20, we get the minimum results, as shown in Fig. 4.2.4b.

5

Simulation Framework

For the implementation and evaluation of the algorithms, a simulation framework was developed in Python 3.7. First, the main structure of the environment is described, and then the test scenarios used in the simulations are presented.

5.1 Design and Functionality

The basis of the environment is to create the problem instance, which is an RBS configuration and a carrier set. This can be generated either deterministically or randomly. A basic frame model associated with the given RBS configuration is then created, before running the algorithms on the problem instance as shown in Figure 5.1.1.

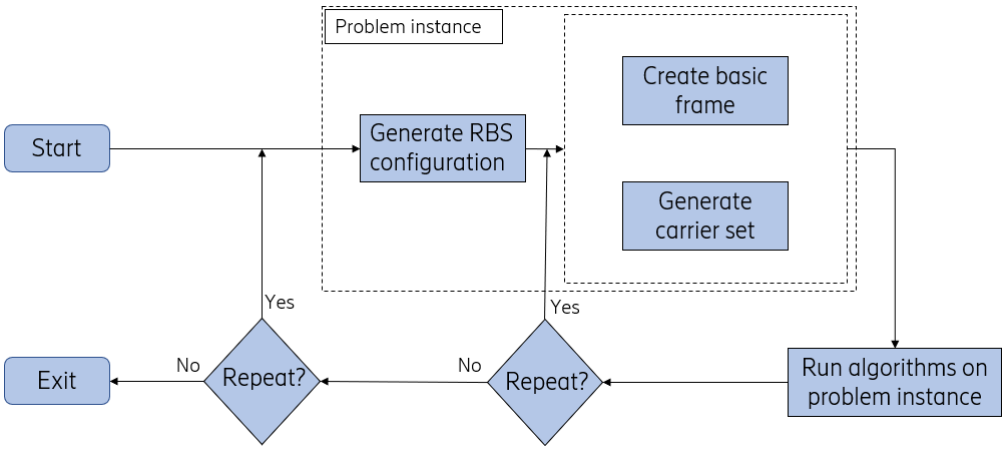


Figure 5.1.1: *Program flow for running algorithms on a problem instance.*

The modelling of the problem instance was described in Chapter 2. To evaluate the algorithms, problem instances were generated in different scenarios by varying the parameters needed to create them. The available units for the RBS configurations in the simulation are shown in Table 5.1.1.

Table 5.1.1: *The available REs, defined by their supported IQ-formats, and line bit rates for generation of RBS configurations.*

Parameter	**Available units**
REs	20, 24, 30, 20/24, 24/30, 20/24/30 [bits]
Line bit rates	2.5, 4.9, 9.8, 10.1 [Gbps]

In order to generate a random problem instance using the available units from Table 5.1.1, the following parameters are needed:

- Number of RE nodes
- Number of carriers
- Max AxC-slots per carrier
- Number of unique line bit rates

The two main categories of algorithms that was tested was the IQ-selection algorithms for minimizing subframes and the allocation algorithms in Chapter 3 and 4, respectively.

5.2 Test Scenarios for Minimizing Subframes

As we saw in Chaper 3, we are not particularly interested in the actual allocation onto the basic frame for evaluating these algorithms. Thus, the basic frame creation was skipped for these simulations. The result of interest is the minimum number of subframes that can be used for a solution to a problem instance. This is evaluated by looking at how often the algorithm finds the optimal solution in terms of subframes, and how many subframes that each algorithm uses on average. In addition, the number of used bits is also evaluated.

The test scenarios were designed to see how the size parameters, with an exception for the line bit rates, for creating the problem instances affects the performance of the algorithms.

5.2.1 IQ-selection Test scenario 1: Varying Configuration Size

In the first test scenario, the aim is to see how the configuration size affects the different algorithms. Since there are in total seven different types of REs used for the simulations (Table 5.1.1), the largest RBS configuration possible is seven for these simulations. As the line bit rates are not considered, multiple REs of the same type is equivalent to a single RE from the view of these algorithms. Therefore, the size of the RBS configuration is defined as how many types of REs that are in the configuration for these simulations. Regarding the other parameters, they are randomly chosen from their available ranges for each problem instance. The simulation was implemented according to the steps described in Alg. 9.

Algorithm 9 IQ-selection test scenario 1: varying configuration size

for RE COUNT $\leftarrow$ 1 to 7 **do**
  Create a configuration of RE COUNT unique RE types.
  $n \leftarrow$ random carrier set size
  for $i \leftarrow$ 1 to n **do**
    AXC SLOTS $\leftarrow$ random AxC-slot size
    DEST RE $\leftarrow$ random RE from configuration
    Create carrier, c_i, using the values AXC SLOTS and DEST RE.
  end for
  Run IQ-selection algorithms on the created configuration and carrier set.
end for

5.2.2 IQ-selection Test Scenario 2: Varying Carrier Set Size

In this test, described in Alg. 10, the number of carriers in each problem instance is varied. The other parameters are chosen randomly.

Algorithm 10 Test scenario 2: varying carrier set size

```
for CARRIER COUNT ← 1 to 15 do
  RE COUNT ← random configuration size in interval [1, 7].
  Create a configuration of RE COUNT unique RE types.
  n ← random carrier set size
  for i ← 1 to CARRIER COUNT do
    AXC SLOTS ← random AxC-slot size
    DEST RE ← random RE from configuration
    Create carrier, c_i, using the values AXC SLOTS and DEST RE.
  end for
  Run IQ-selection algorithms on the created configuration and carrier set.
end for
```

5.2.3 IQ-selection Test Scenario 3: Varying Maximum AxC-slots per Carrier

For the last test, described in Alg. 11, the maximum AxC-slot size of each carrier is varied. This means that the number of AxC-slots for each carriers is still randomly generated, but we look at how the size of the set from which it is randomly generated affects the performance. The available sizes was chosen in order to get a fairly realistic distribution, since it is currently more common with smaller sizes.

Algorithm 11 IQ-selection test scenario 3: varying max AxC-slots per carrier

```
for MAX AXC-SLOTS ∈ {1, 2, 3, 4, 5, 6, 8, 10, 15, 20, 30} do
  RE COUNT ← random configuration size in interval [1, 7].
  Create a configuration of RE COUNT unique RE types.
  n ← random carrier set size
  for i ← 1 to CARRIER COUNT do
    AXC SLOTS ← random AxC-slot size from MAX AXC-SLOTS
    DEST RE ← random RE from configuration
    Create carrier, c_i, using the values AXC SLOTS and DEST RE.
  end for
  Run IQ-selection algorithms on the created configuration and carrier set.
end for
```

5.3 Test Scenarios for Allocation Algorithms

The allocation algorithms were evaluated in a different way than the IQ-format selection algorithms in the previous section. However, since the IQ-format selection must be done before the allocation, we cannot neglect the first step. Thus, these tests are for the entire allocation process, including both IQ-format selection and allocation. The algorithm combination is denoted as *'allocation algorithm - iq-selection algorithm'*. For example, if the IQ-format is selected using MF and the allocation is done using HFF, this is denoted as 'HFF-MF'. The parameters of interest was to look at the percentage of successful allocations and the average number of allocated slots. For the first parameter, a decision

problem whether an algorithm successfully manages to allocate all carriers for a given problem instance was formulated. This means that the test runs a bit differently from the ones in Section 5.2. First of all, line bit rates were included, and a basic frame was created for the configuration. The size of the basic frame was modelled after the highest line bit rate in the configuration. Regarding the carrier sets used when evaluating the amount of successful allocation, we can only see any impact if they are large enough relative to the capacity of the basic frame. If the load on the basic frame is too low, then allocation is possible using basically any allocation algorithm. However, the carrier set must of course be theoretically possible to fit in the basic frame. That is, for each subframe region, the total number of bits needed for the carriers cannot exceed the capacity of the subframes.

In order to evaluate the amount of successful allocations, a data set of critical carriers, i.e. carrier sets considered to be of interest, were generated. A carrier set was defined as critical when at least one of the evaluated algorithms was not able to successfully allocate all its carriers. In order to find critical carrier sets, they were generated iteratively as shown in Alg. 12.

Algorithm 12 Generation of critical carrier sets

```
CRITICAL SET ← ∅
Generate configuration
while true do
  Generate a random carrier which theoretically fits in the basic frame and add to
  carrier set.
  if no theoretical possible set is found then
    Add the latest valid carrier set to CRITICAL SET if it has not been added yet.
    return
  end if
  Run algorithms on the carrier set.
  if all algorithms fails then
    Add set to CRITICAL SET.
    return
  end if
  if at least one algorithm fails the allocation then
    Add set to CRITICAL SET.
  end if
end while
```

First of all, an RBS configuration is generated after the desirable specifications for the simulation in question, with the related basic frame. Then, a carrier set is generated one carrier at a time. For each carrier, its destination RE and size is randomly generated. However, it cannot exceed the theoretical limit of the basic frame. For example, if there is theoretically only one AxC-slot left in a basic frame, then the generated carrier cannot have a size of more than one AxC-slot. The basic frame is considered theoretically full when the minimum number of used bits for all carriers reaches the capacity of the basic frame.

When a carrier has been generated, it is added to the carrier set. The algorithms then runs on the problem instance, trying to allocate the carriers to the basic frame. If all algorithms manages to allocate all carriers in the set, the carrier set is considered to be non-critical. A new carrier is then added to the carrier set, in the same way as before. All algorithms then

run from scratch again. Then the new carrier set is treated as completely new, and the basic frame is cleared before the allocation. When a carrier set is found, such that at least one algorithm fails to allocate, the carrier set is considered to be critical. The algorithms was then evaluated by trying to allocate randomly chosen critical carrier sets.

Due to its bad performance, LFF-MF and LFF-FC were not used when generating the critical carrier sets. The reason for this was to keep the amount of too simple carrier sets down. However, the performances of LFF-MF and LFF-FC were still evaluated, in order to see the differences compared to SA/LFF.

To measure the average maximum number of allocated slots for each algorithm, carrier sets were generated iteratively in a similar way, according to Alg. 13.

Algorithm 13 Finding the maximum capacity of an allocation algorithm.

```
MAX ALLOCATED SLOTS ← 0
Generate configuration
while true do
  Generate a random carrier which theoretically fits in the basic frame and add to
  carrier set.
  if no theoretical possible set is found then
    return MAX ALLOCATED SLOTS
  end if
  Run algorithms on the carrier set.
  if an algorithm successfully allocated all carriers then
    Update MAX ALLOCATED SLOTS
  else
    return
  end if
end while
```

For each generated carrier set, the evaluated algorithm tries to allocate all carriers. The carrier set continues to be expanded until the algorithm fails to allocate it. Then, the amount of slots for the last successful carrier set is returned as the maximum number of allocated slots for the algorithm.

In terms of varying generation parameters we look at the configuration size, the max number of AxC-slots per carrier and the number of unique line bit rates in the configuration.

5.3.1 Allocation Test Scenario 1: Varying Configuration Size

As in the IQ-selection tests in Section 5.2, we look at how the configuration size affects the algorithms. Since the line bit rates are included in these tests, we are not only interested in unique RE types. If two REs of the same type are using different line bit rates, they will affect the allocation differently, and are therefore considered interesting.

Algorithm 14 Allocation Test Scenario 1: Varying Configuration Size

for RE COUNT $\leftarrow$ 1 to 7 **do**
  Create a configuration of RE COUNT unique RE types with randomly chosen line bit rates.
  Run Alg. 12
end for

5.3.2 Allocation Test Scenario 2: Varying Unique Line Bit Rates

In this test, we look at how the number of unique line bit rates in the configuration affects the algorithms. As with the previous test, it can also be seen as an indication on how complex the configuration is. The maximum number of unique line bit rates that are considered is four, see Table 5.1.1. In terms of configuration size, it is randomly chosen. However, it can of course not be less than the number of unique line bit rates.

Algorithm 15 Allocation test scenario 2: varying unique line bit rates

for LINERATE COUNT $\leftarrow$ 1 to 4 **do**
  LINERATES $\leftarrow$ LINERATE COUNT number of line bit rates.
  Create a configuration of RE COUNT $\geq$ LINERATE COUNT number of REs, with line bit rates from LINERATES
  Run Alg. 12
end for

5.3.3 Evaluated Algorithms

The allocation algorithms that were tested are the one presented in Chapter 4. However, the offline algorithms SA and RB can be run with different IQ-selection algorithms. How these perform in combination with the allocation algorithms was evaluated as well. The allocation algorithms that were tested are shown in Table 5.3.1. Each were evaluated for both MF and FC IQ-selection algorithms. The OPT IQ-selection was not used for these simulations, due to complexity issues. All RBS configurations were also assumed to have non-decreasing line bit rates.

Table 5.3.1: *The allocation algorithm combinations that were evaluated. Each allocation algorithm was combined with both MF and FC for IQ-selection.*

Allocation algorithms
High First-Fit (HFF)
Sorting Algorithm/High First-Fit (SA/HFF)
Low First-Fit (LFF)
Sorting Algorithm/Low First-Fit (SA/LFF)
Combined First-Fit (CFF)
Sorting Algorithm/Combined First-Fit (SA/CFF)
Region-Based (RB)

6

Results

In this chapter, the results for the test simulations presented in Chapter 5 are presented.

6.1 Minimizing Number of Subframes Results

First the results for the test scenarios presented in Section 5.2 for evaluating the MF and FC algorithms according to Chapter 3.

6.1.1 IQ-selection Test Scenario 1: Varying Configuration Size

The results for the comparison between OPT, MF and FC for different configuration sizes are shown in Fig. 6.1.1.

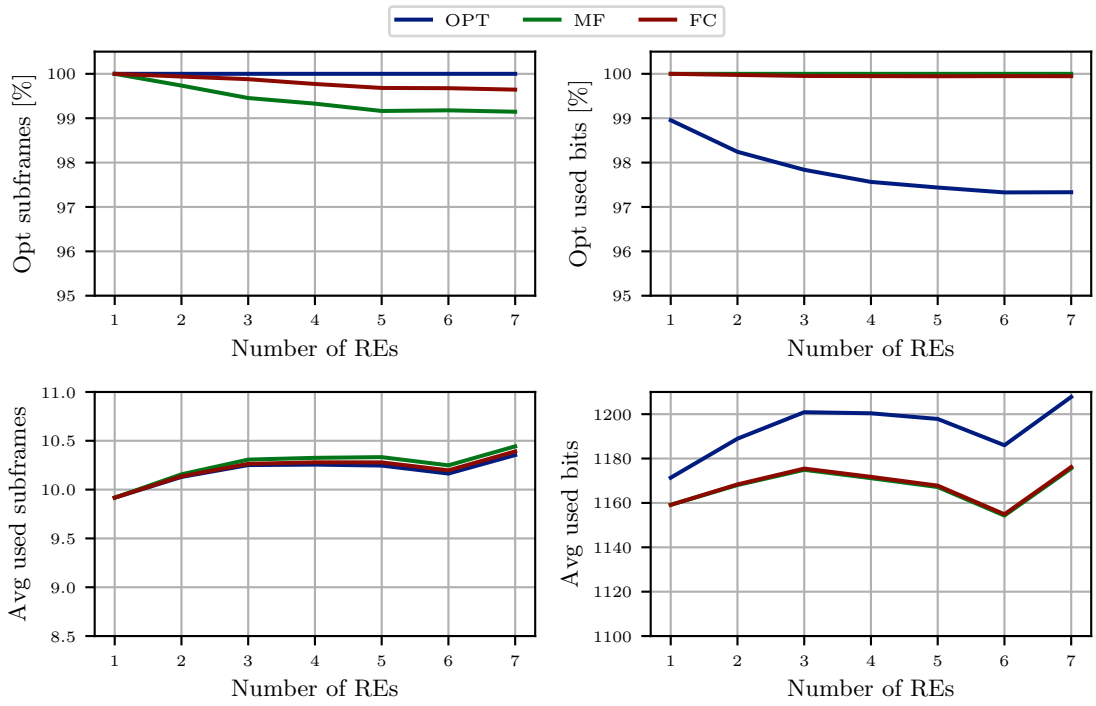


Figure 6.1.1: *Testing IQ-selection algorithms for different configuration sizes.*

In terms of minimizing the minimum number of used subframes, both MF and FC performs further away from OPT for larger configuration sizes. The biggest difference is for configurations with seven REs, where MF and FC are 1% and 0.2% from optimum

respectively. Looking at the average used subframes per carrier set, MF uses about 0.1 subframes more than OPT on average for the largest configuration size. Regarding the number of used bits, it is obvious that the optimum solution in terms of subframes is not to use as few bits as possible. Note that the algorithm names are a bit misleading here, as OPT refers to optimal in terms of minimizing the number of subframes (see Section 3.3.1), and that MF is the optimal algorithm in terms of minimizing the number of used bits. For more complex configurations, the optimal solution in terms of used bits gets further away from the minimal solution, which correlates with the result that MF performs worse for larger configuration.

6.1.2 IQ-selection Test Scenario 2: Varying Carrier Set Size

In Fig. 6.1.2, different carrier set sizes are analyzed.

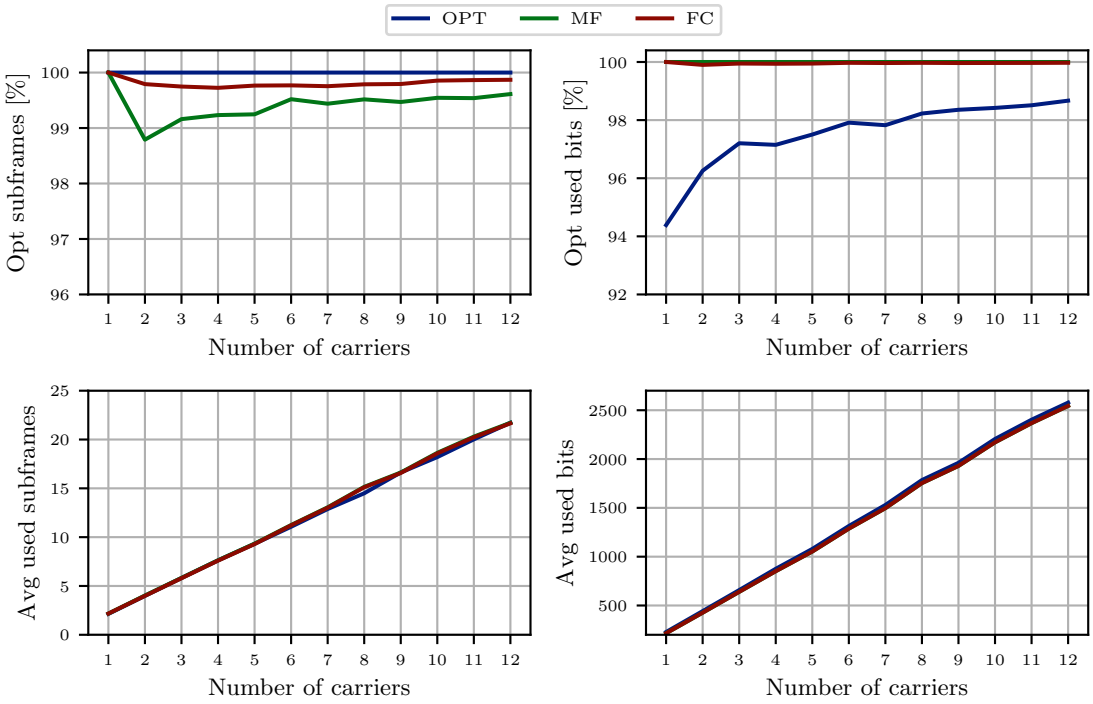


Figure 6.1.2: *Testing IQ-selection algorithms for different carrier set sizes.*

Except for the case with only one carrier, the minimal and optimal solutions approaches each other for larger carrier sets. This is also true for FC, although the differences are not as significant. In the two lower graphs, we can see how the carrier set size increases, but the differences are not clearly visible. Looking at the values in Table A.1 from Appendix A, the difference in terms of subframes is about 0.1 subframes per carrier set between MF and OPT for all set sizes. Thus, the difference reduces in terms of percentage for larger carrier sets.

6.1.3 IQ-selection Test Scenario 3: Varying Maximum AxC-slots per Carrier

The results for comparisons of different carrier sizes, i.e. max number of AxC-slots per carrier, are shown in Fig. 6.1.3.

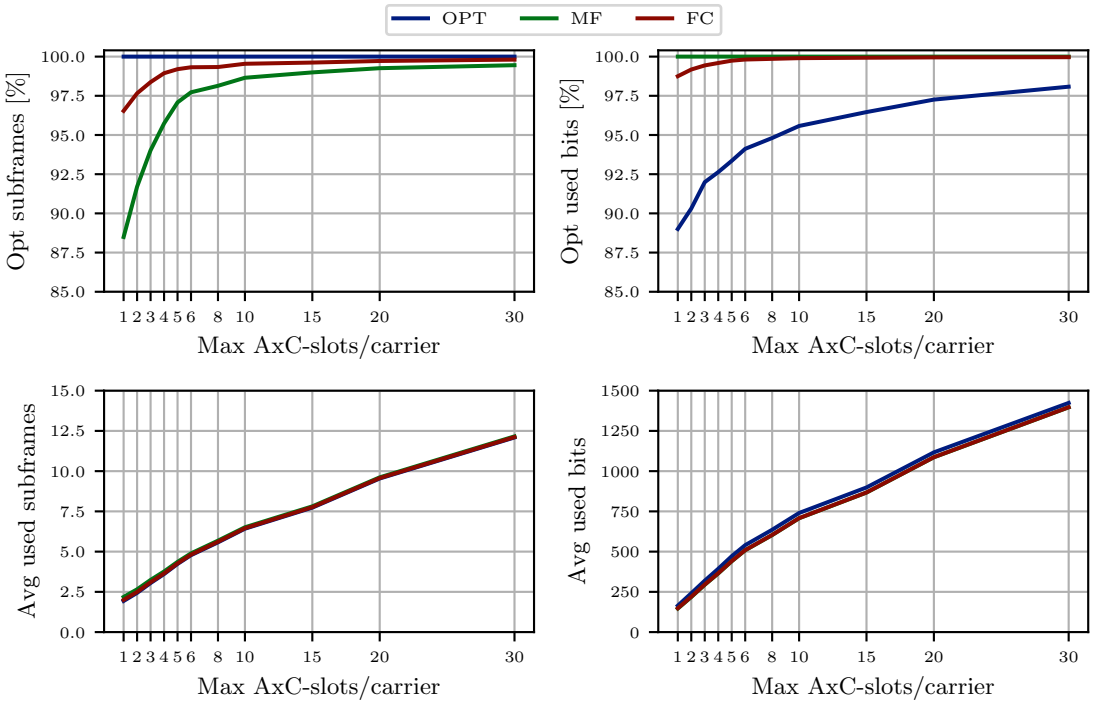


Figure 6.1.3: *Testing IQ-selection algorithms for different values of max AxC-slots per carrier.*

The trend is quite similar to the carrier set size variation, with reduced difference for larger carrier sizes.

6.2 Allocation Strategy Results

In this section, we look at full allocation algorithm solutions, including IQ-selection and allocation onto a basic frame. First, the four allocation strategies HFF, LFF, CFF and RB are examined internally, to see how IQ-selection algorithms and offline sorting algorithms (SA) affects the performance for the different test scenarios. Then the offline and online alternatives are compared towards each other.

6.2.1 HFF

The simulation results for different modifications of the HFF algorithm are shown for different configuration sizes in Fig. 6.2.1.

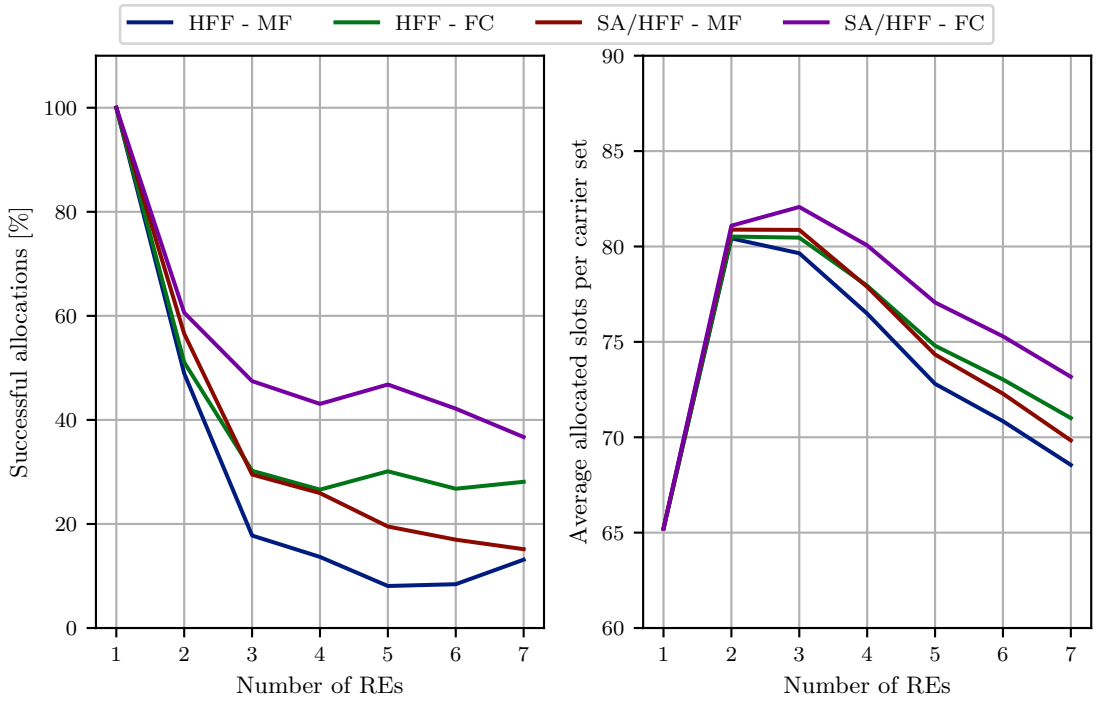


Figure 6.2.1: *Testing HFF allocation algorithms with MF and FC IQ-selection for different configuration sizes.*

The differences between the four HFF implementations gets more significant for larger configuration sizes. The biggest effect from the offline implementations is when five REs are used, where SA/HFF-FC manages to successfully allocate 46.8 % of the critical carrier sets, compared to 8.1 % for the online HFF-MF. For this case, SA/HFF-FC manages to allocate 4.3 more AxC-slots per carrier set on average than HFF-MF. For configurations with up to four REs, the effect of using FC or SA is similar with a small advantage for SA. However, for five or more REs, the improvement of using only FC is bigger than only sorting.

The next test scenario, with different line bit rates, is shown in Fig. 6.2.2.

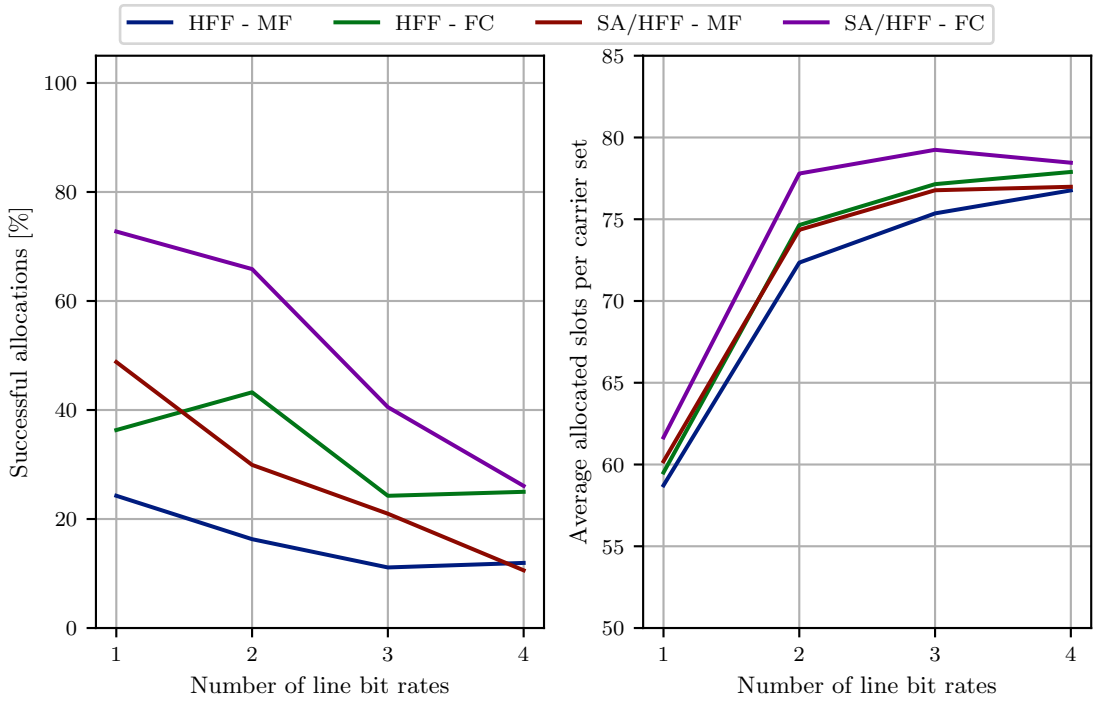


Figure 6.2.2: *Testing HFF allocation algorithms with MF and FC IQ-selection for different number of unique line bit rates.*

In terms of successful allocations the differences decrease for many different line bit rates. In the case of one line bit rate, SA/HFF-FC successfully allocates 48.5 percentages more sets than HFF-MF, while for four line bit rates, the difference is 14.1 percentages. In terms of allocated slots, the difference for these cases are 2.9 and 1.7 slots respectively.

For the case of varying the number of REs, only using FC has more effect than only sorting for an increasing number of line bit rates. When there are four different line bit rates in the configuration, the effect of sorting is non-existing.

The best offline alternative between the HFF implementations was SA/HFF-FC, i.e. using both offline sorting and FC.

6.2.2 LFF

For LFF, the difference for sorting and non-sorting implementations is very clear, as seen in Fig. 6.2.3.

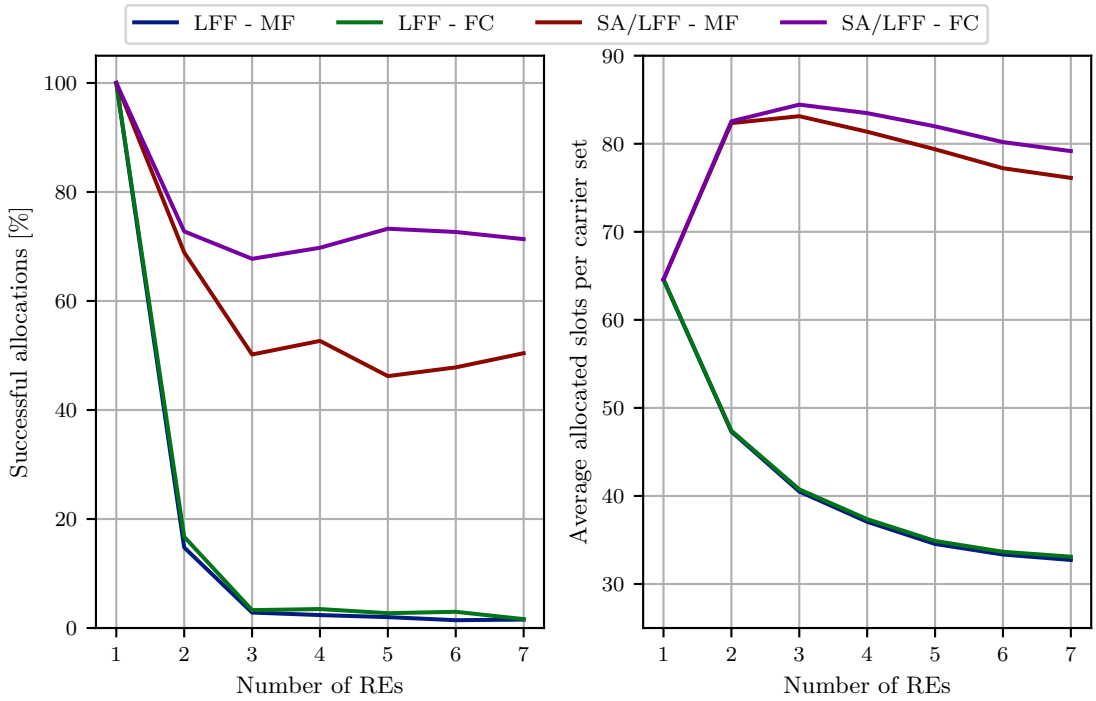


Figure 6.2.3: *Testing LFF allocation algorithms with MF and FC IQ-selection for different configuration sizes.*

Unlike HFF, the most improvement for offline implementations are due to sorting, especially for large configurations. LFF without sorting performs extremely bad for all cases but the single RE. We can still see that although the biggest improvement is achieved by sorting, there is still a significant effect of using FC with sorting compared to MF with sorting, where SA/LFF-FC manages to successfully allocate 27 percentages more carrier sets than SA/LFF-MF. The same goes for the performance when comparing different line bit rates in Fig. 6.2.4.

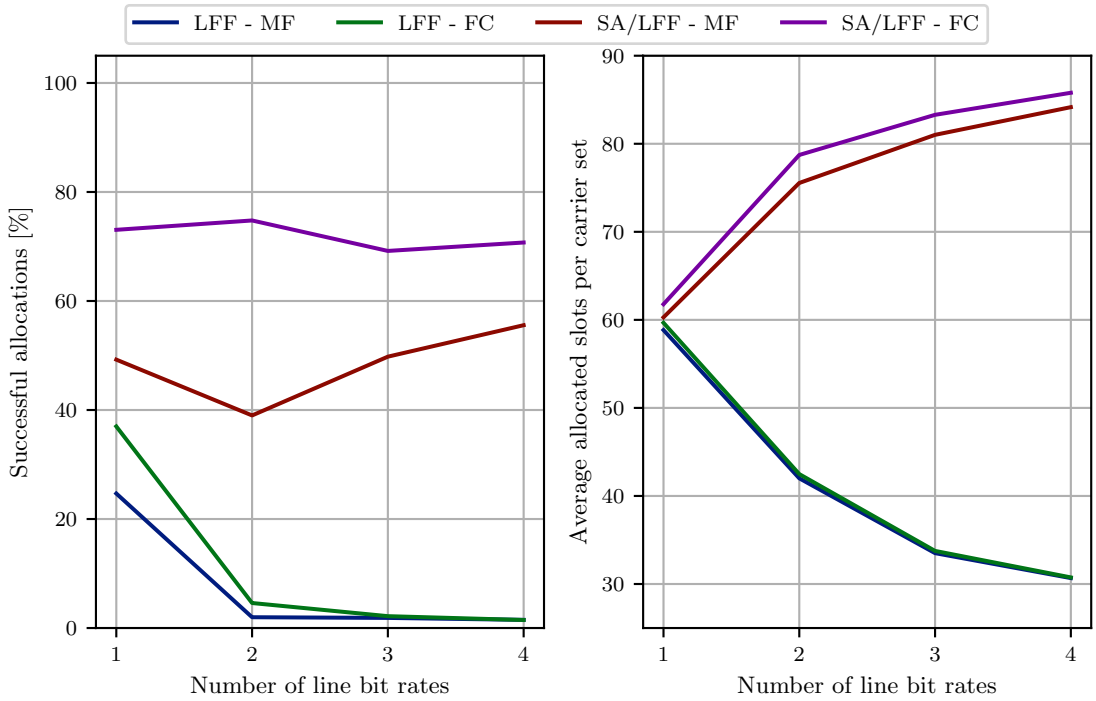


Figure 6.2.4: *Testing LFF allocation algorithms with MF and FC IQ-selection for different number of unique line bit rates.*

The only exception is for the case of one line bit rate, where sorting does not have as much effect as before. Still, SA/LFF-FC performs way better than the other alternatives. SA/LFF-FC was chosen as the best LFF alternatives, with very stable results independently of the configuration size.

6.2.3 CFF

For CFF, the trend is similar to that of HFF, with more significant differences for larger configurations sets, as can be seen in Fig. 6.2.5. The online version CFF-MF keeps up with the rest for up to two REs.

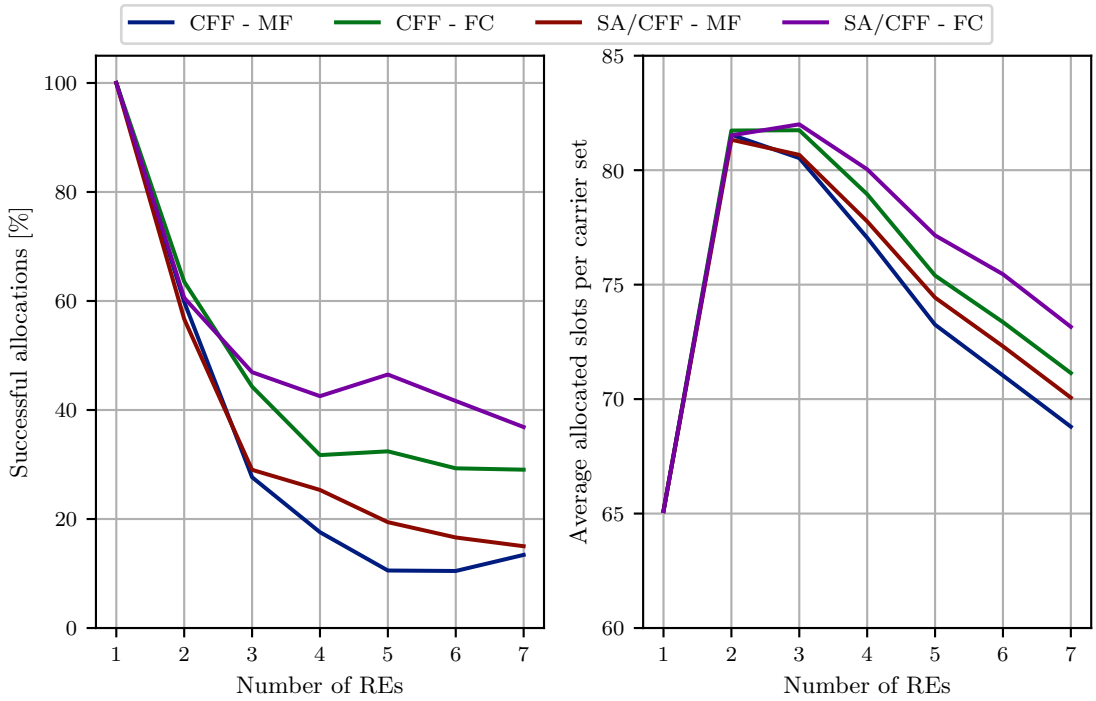


Figure 6.2.5: *Testing CFF allocation algorithms with MF and FC IQ-selection for different configuration sizes.*

For different line bit rates, in Fig. 6.2.6, we can once again see that the differences is reduced, or at least constant, for more complex configurations.

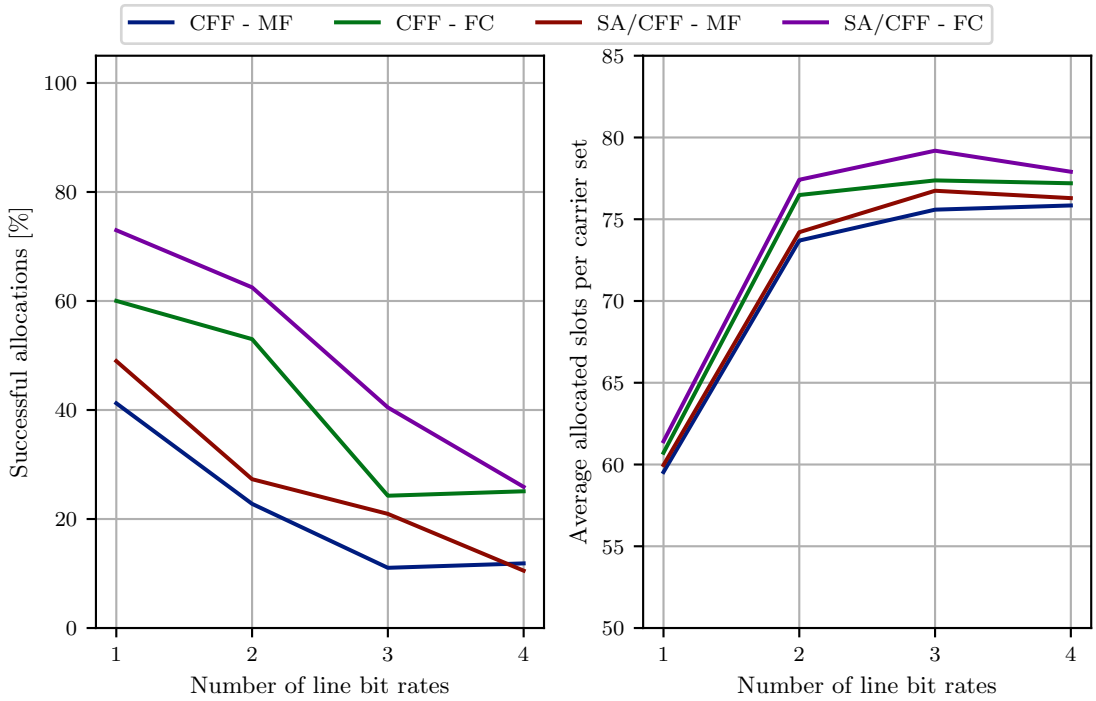


Figure 6.2.6: *Testing CFF allocation algorithms with MF and FC IQ-selection for different number of unique line bit rates.*

Again, sorting is not as effective as FC, although the best alternative is to used both as in SA/CFF-FC. Note that for more than two line bit rates, CFF is equivalent to HFF.

6.2.4 RB

For the RB algorithm, the differences between the two IQ-selection strategies are shown for different configuration sizes in Fig. 6.2.7.

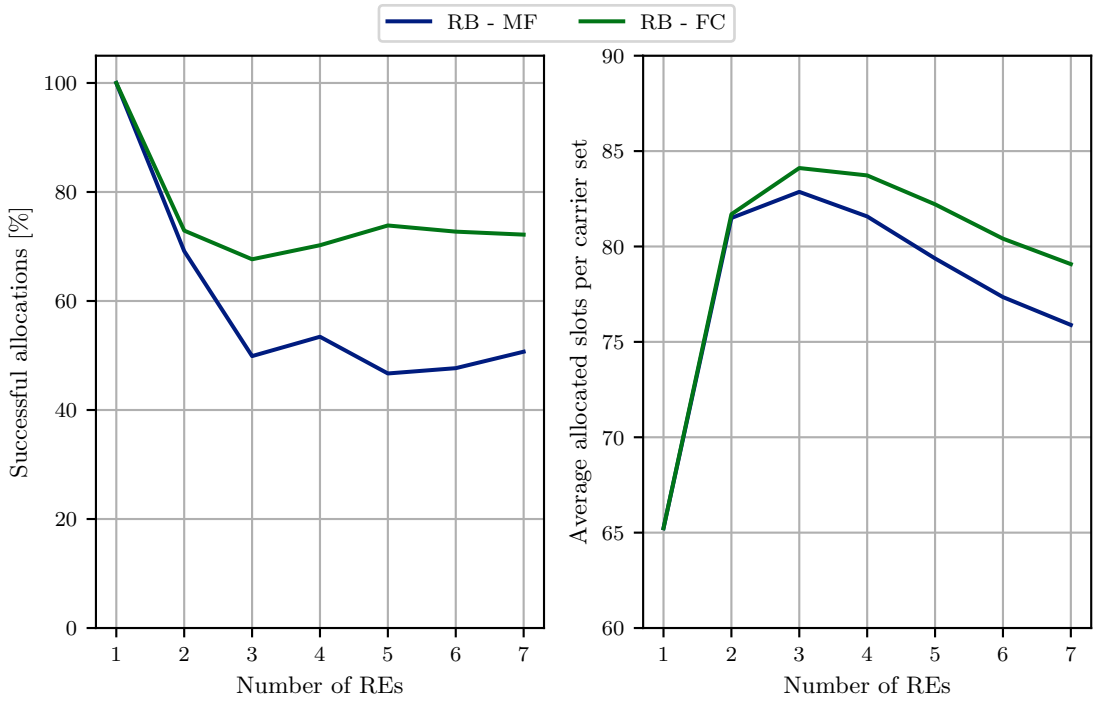


Figure 6.2.7: *Testing RB allocation algorithms with MF and FC IQ-selection for different configuration sizes.*

Using FC instead of MF with RB gives better results for larger configuration sizes, with 3.2 more slots per carrier set for the case of seven REs. For up to two REs, the difference is very small. In terms of different line bit rates, shown in Fig. 6.2.8, the difference is reduced for higher number of line bit rates, peaking at two different line bit rates.

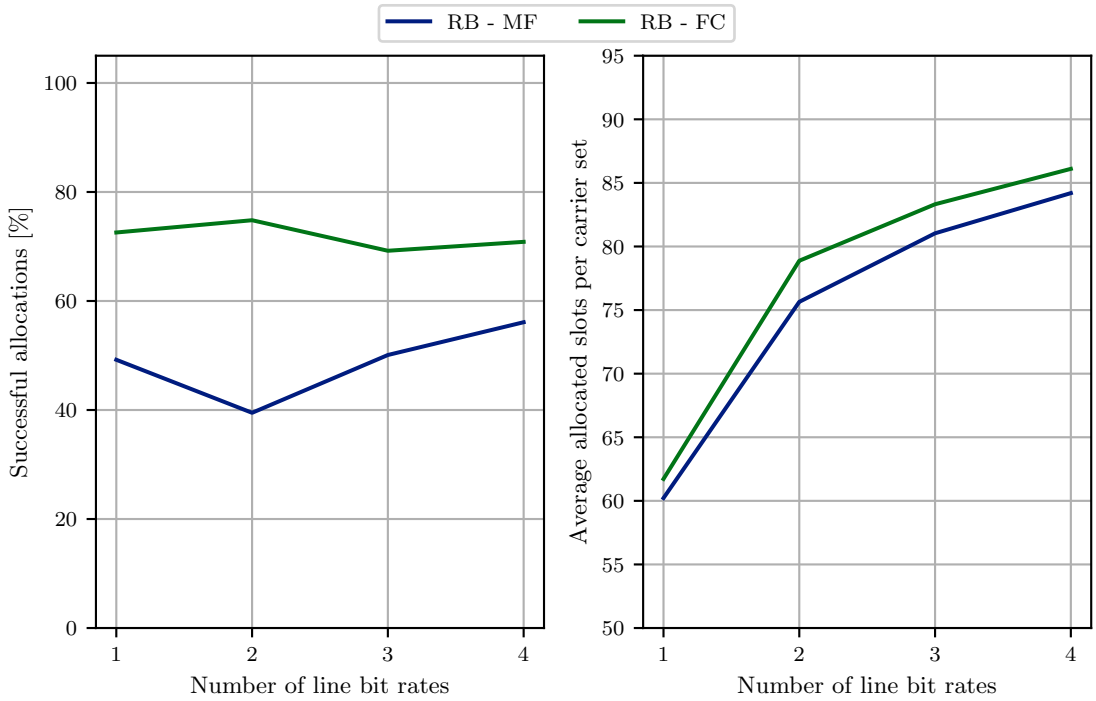


Figure 6.2.8: *Testing RB allocation algorithms with MF and FC IQ-selection for different number of unique line bit rates.*

As RB-FC performs better than RB-MF in every examined scenario, it is clear that FC can be used to improve the RB algorithm. RB is stable and manages to handle different configurations without dropping too much in performance.

6.2.5 Online Comparisons

In Fig. 6.2.9 and 6.2.10, the two best online implementations, HFF-MF and CFF-MF, are compared.

For the online implementations in Fig. 6.2.9 and Fig. 6.2.10, CFF performs performs slightly better for smaller configuration sizes, with at most around 0.8 more slots per carrier set for the case of three REs. Considering the line bit rates, CFF has some advantage over HFF for line bit rates below three (for line bit rates of three and above, CFF and HFF are identical, see Section 4.2.2).

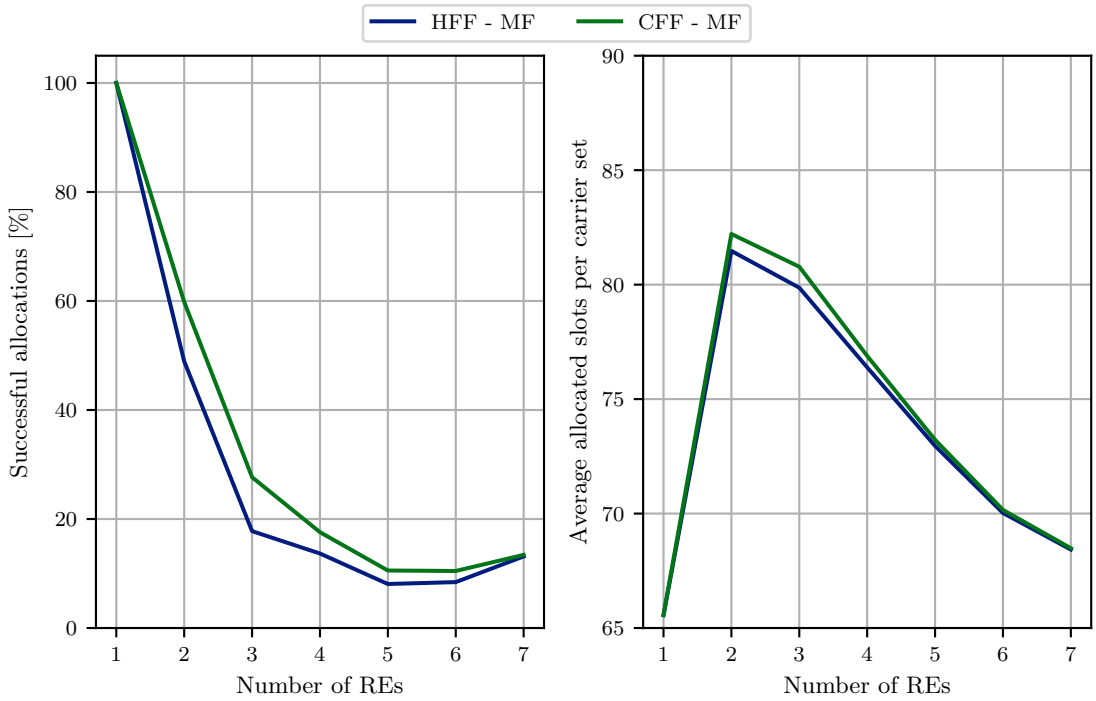


Figure 6.2.9: *Comparing the different online solutions for different configuration sizes.*

We can see that CFF-MF gives slightly better results for most configurations, except very small and very large, where they are equivalent.

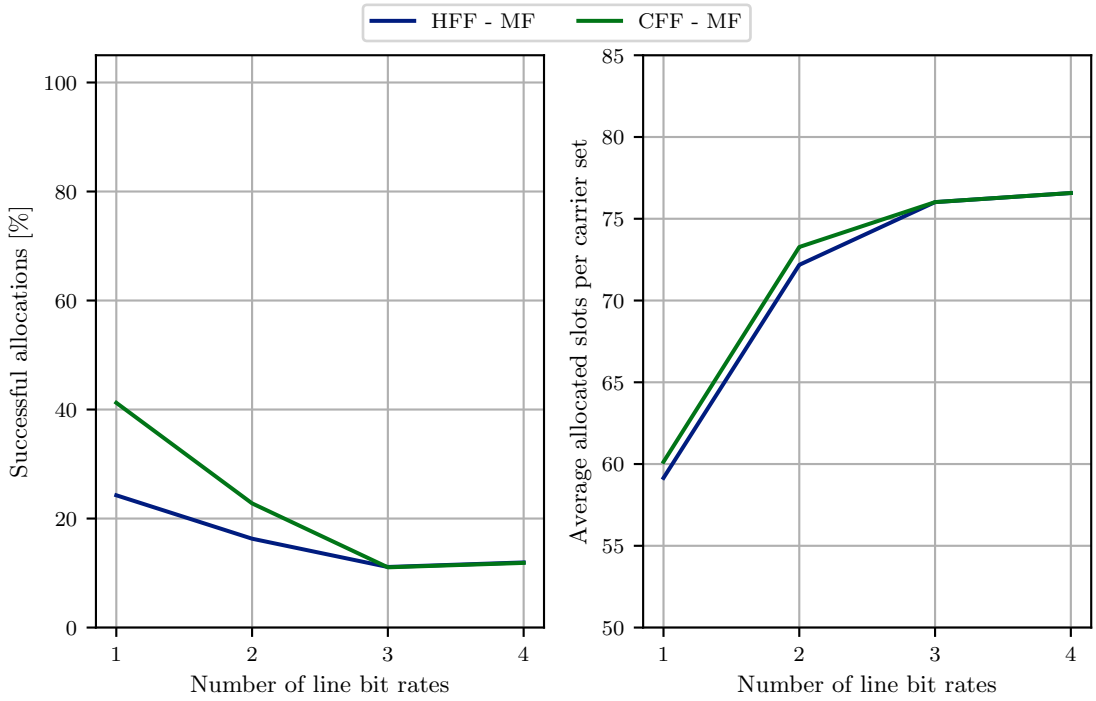


Figure 6.2.10: *Comparing the different online solutions for different number of unique line bit rates.*

In terms of line bit rates, CFF-MF is to prefer for up to two different line bit rates, while they are equivalent for more than that. Considering all scenarios, CFF-MF was chosen as the best-performing online algorithm.

6.2.6 Offline Comparisons

The four most promising offline alternatives based on the previous results, are presented in Fig. 6.2.11 and 6.2.12.

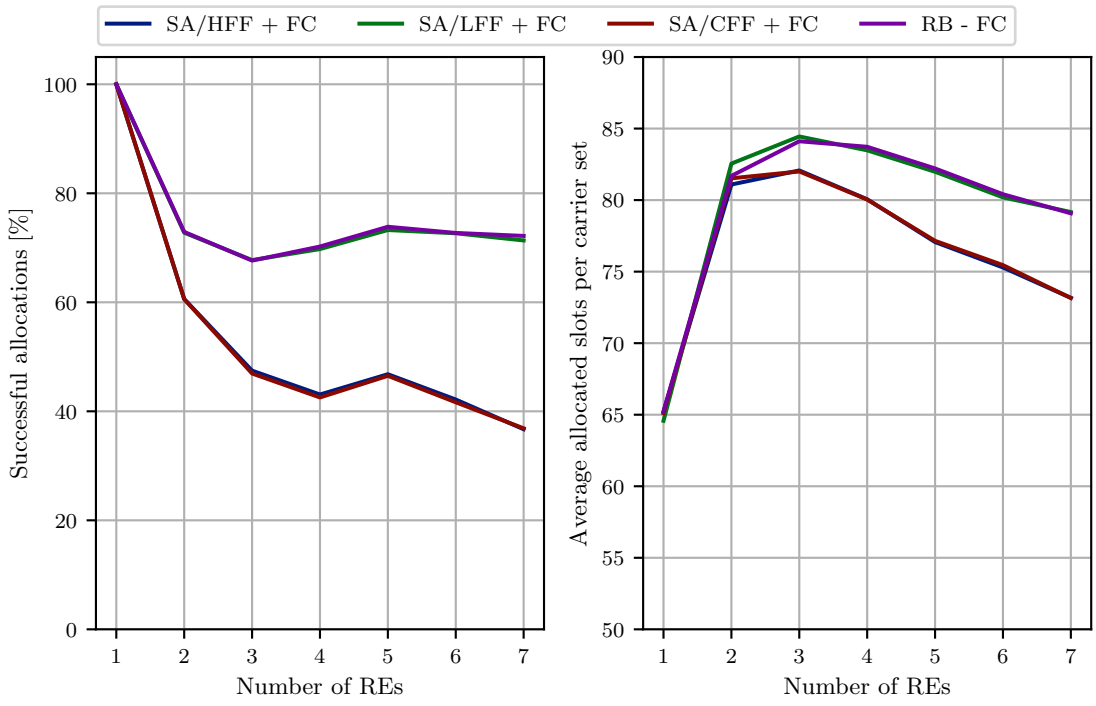


Figure 6.2.11: *Comparing the different offline solutions for different configuration sizes.*

RB-FC and SA/LFF-FC are almost identical in terms of performance, while the other alternatives lack a bit especially for larger configurations.

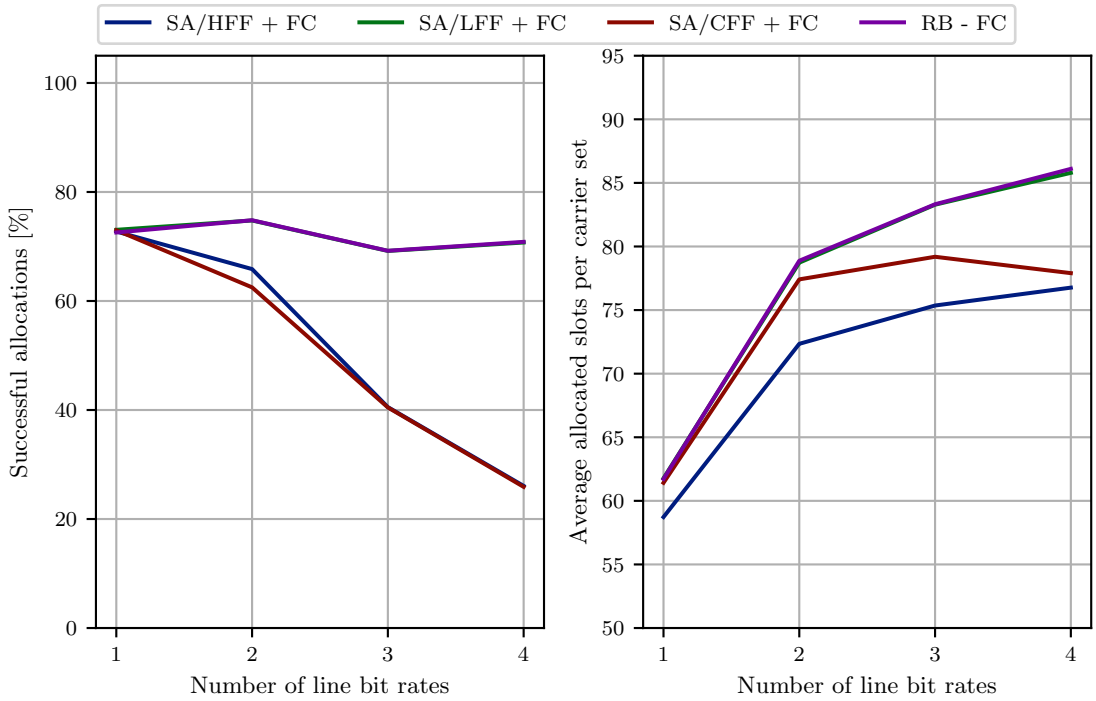


Figure 6.2.12: *Comparing the different offline solutions for different number of unique line bit rates.*

The same goes for different line bit rates, where RB-FC and SA/LFF-FC are superior for many different line bit rates. For the case of two line bit rates, SA/HFF-FC performs better than SA/CFF-FC in terms of successful allocations, but worse in terms of average allocated slots.

6.2.7 Online and Offline comparisons

Finally, the most promising online alternative, CFF-MF, is compared with the three most promising offline alternatives in Fig. 6.2.13 and 6.2.14.

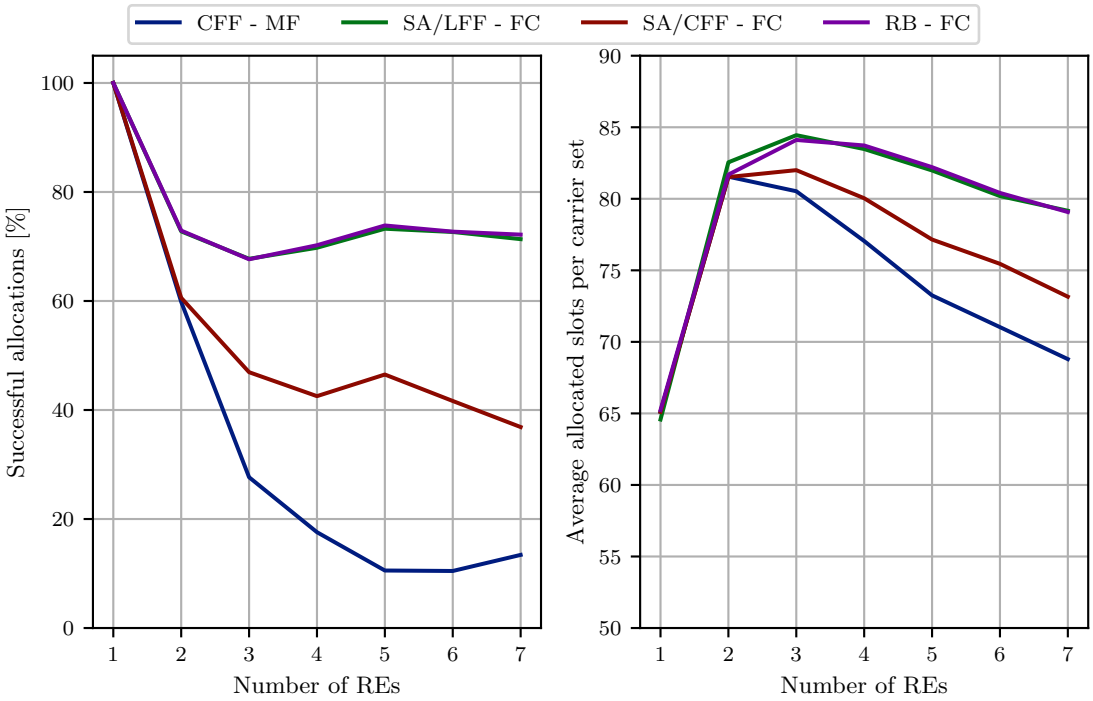


Figure 6.2.13: *Comparing the most promising online and offline solutions for different configuration sizes.*

In terms of configuration size, it is clear that it is for the larger sizes that offline algorithms provides the most benefits. For the case of five REs, RB-FC and SA/LFF-FC successfully allocate about 62 percentages more of the critical carrier sets than CFF-MF. In terms of allocated slots, the difference is about 9 slots per carrier set.

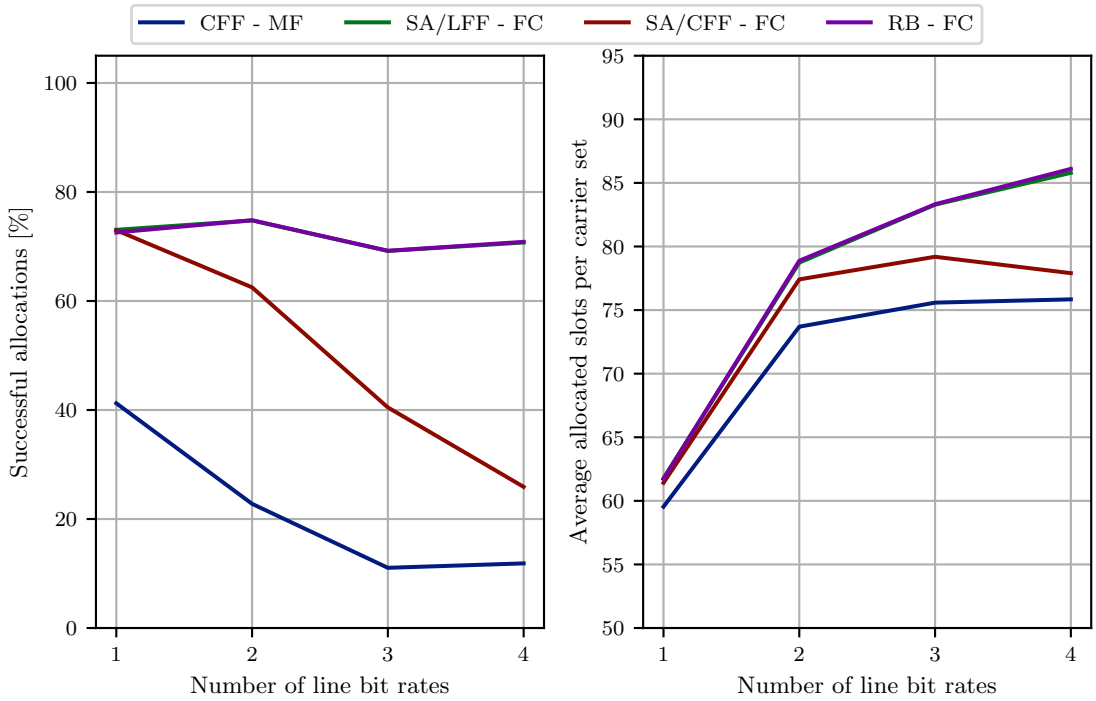


Figure 6.2.14: *Comparing the most promising online and offline solutions for different configuration sizes.*

Finally, we can see that for different line bit rates, there are some significant differences between the online and offline algorithms. For the case of two different line bit rates, RB-FC and SA/LFF are able to allocate 5.2 more slots per carrier set than CFF, and for the case of four different line bit rates, 10.3 more slots per set.

6.2.8 Fragmentation Comparison

We have seen that there is potential for improving the performance by processing the carriers offline. All implementations have in common that the combination of doing the IQ-selection offline with FC and doing either SA or RB for the allocation gives the best results. How much of this improvement that is due to the IQ-selection and due to the allocation strategy seems to depend on the configuration. If we look at the average allocated slots for CFF in Fig. 6.2.6, we can see that for less than three different line bit rates, CFF - FC performs better than SA/CFF - MF, which indicates that the IQ-selection has more effect than the sorting. For three and four line bit rates, the difference is reduced.

In Fig. 6.2.15, the average improvement in terms of allocated slots for RB compared to CFF is shown. CFF is compared to RB with both MF and FC, in order to see how much of the improvement that depends on each of the fragmentation types.

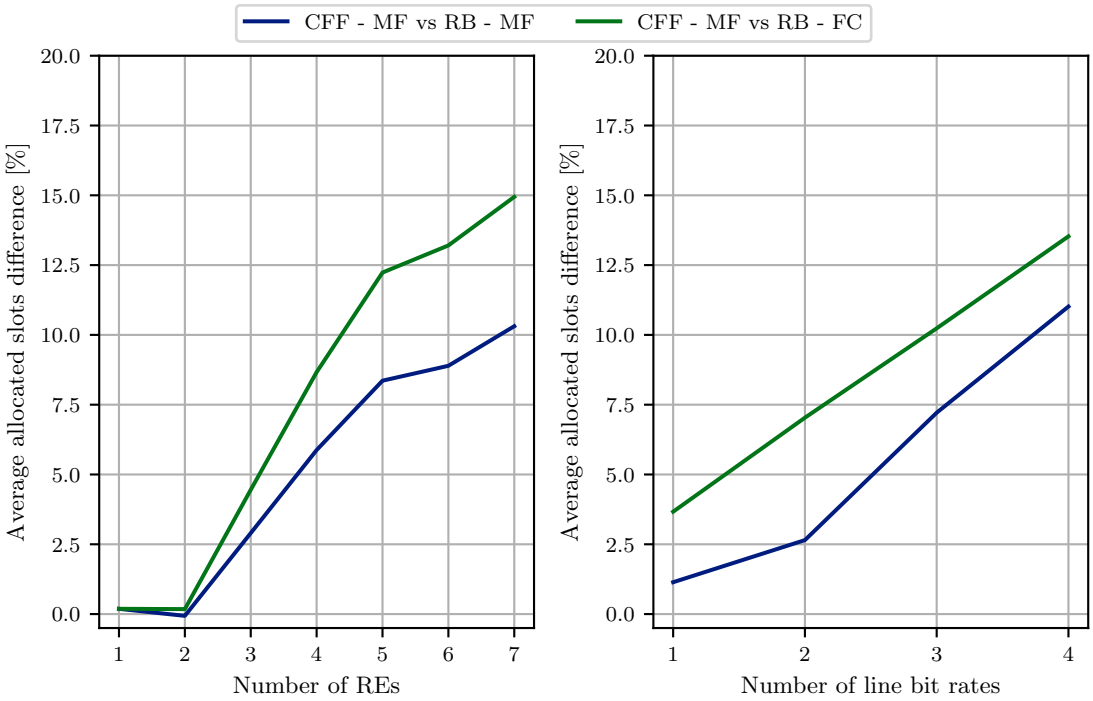


Figure 6.2.15: *The average improvement for RB compared to CFF with MF and FC.*

The benefits from RB increases with both the configuration size and number of line bit rates. The biggest difference is for the case of seven REs, where the average number of allocated slots can be improved by 15 % if using RB - FC instead of CFF - MF. When comparing the effects of MF and FC, we can see that for the varying configuration size, the change from MF to FC stands for about one third of the overall improvement, except for very small configurations. In terms of line bit rate, the significance of the IQ-selection change is the highest in the case of two different line bit rates, with more than half of the overall improvement. For the case of four different line bit rates, the contribution from FC is much smaller, with only about 9 % of the overall improvement.

7

Discussion

In this chapter, the results presented in Chapter 6 are discussed.

7.1 IQ-Selection Algorithms for Minimizing the Number of Subframes

The key take-aways from the results in Section 6.1 are that increasing configuration size makes it harder for MF and FC to minimize the number subframes, while the fragmentation is quite constant for different carrier set sizes. However, the larger the carrier set is, the less effect from the fragmentation can be seen.

Regarding the configuration sizes, it is expected that increasing the number of REs makes it harder to find the optimal solution, as the set of possible solutions increases. Looking at the RE alternatives, REs supporting only one format reduces the complexity of the carrier set enourmously. Likewise, configurations with a common minimal format among all REs also results in the MF as optimal solution. The probability for these cases is of course higher for small configurations, which can explain why MF seems to get further away from OPT for an increased number of RE nodes. The same goes for FC, which is based on MF but with some limited modifications. This can clearly be seen from the optimal used bits, where the difference between MF and FC is nearly constant for all configuration sizes. Thus, we can see FC as an improved version of MF. For even larger configurations, it is reasonable to assume that FC will follow the pattern of MF in terms of optimal subframe rate, but with a constant improvement.

When evaluating these results, it must be taken into consideration how the simulations were designed. The maximum number of AxC-slots was randomly generated for each carrier in this simulation. From the results in Section 6.1.2, we saw that the difference between MF and the other algorithms is larger for low maximum AxC-slot values. Special cases, such as large configurations with low maximum number of AxC-slots are likely to yield a bigger improvement of FC and OPT compared to MF. It is recommended to look further into the case of a more restricted AxC-slot generation for the simulations.

In 5G NR, the channel bandwidth increases which allows for more AxC-slots per carrier. Higher line bit rates in the future also allows for more carriers per set, and the number of REs for each configuration is expected to expand. Thus, the most interesting scenarios to look further into is those where all the investigated parameters are increased.

Another important aspect of the simulation is that it assumes that only one line bit rate is used. For the real implementation, we can have up to four line bit rates (and possibly even more, outside the scope of this thesis). When FC was implemented for the allocation algorithms, the carriers for each line bit rate was divided into subsets. For the case of four

different line bit rates, that means that there are four carrier subsets for which the number of subframes should be minimized. Thus, both the set size and the number of REs shrinks for each subset. Since the difference in average number of subframes seems to be near constant independently of the carrier set size, dividing a set into four subsets, would mean that the difference between the algorithms most likely would increase.

Once again, it should be emphasized that these are the results of the minimum number of used subframes for the solutions of each algorithms, and other fragmentation issues, such as the allocation order of the carriers, are neglected. Thus, these results should rather be seen as a comparisons of the offline implementation of MF and FC, where carriers of the same IQ-format can be allocated consecutively, as in allocation algorithm SA, described in Section 4.2.3.

7.2 Allocation Algorithms

7.2.1 Simulation Framework

Before evaluating the algorithms, we raise some important aspects of the simulation framework that was used.

In terms of the generation of critical carrier sets described in Section 5.3, it is important to note that it depends on the algorithms that are tested in the simulation, as a carrier set is not considered critical unless at least one algorithm has failed to allocate it, or if it is not theoretically possible to allocate more. This has impact on the successful allocations parameter, as it depends on how hard the critical carrier sets are to allocate. If the threshold for considering a carrier set critical is lowered, then the amount of successful allocations increases. As this report aims to compare the presented algorithms to each other, the critical carrier sets were defined based on these. However, for less critical carrier sets, it is likely that several of the well-performing algorithms examined in this thesis performs equally well. In order to generate more challenging carrier sets, only high-performing algorithms should be included in the carrier set generation.

Another aspect of this is how the theoretical maximum is defined. Since we do not have an optimal allocation algorithm to compare with, we cannot prove that there exists a solution such that a carrier set can be allocated, when none of our algorithms manages to allocate it. However, we can prove that a carrier set cannot be allocated if the minimum amount of data that it contains exceeds the capacity of the basic frame. This is the threshold that was used. This means that it is theoretically possible that a critical carrier set is generated, although it does not exist a solution for it. In the end this is a trade-off. The threshold could be lowered, but that would mean that there is a risk that a carrier set is considered non-valid, although it is valid. This would result in overestimated performance values, which is not desirable either. The fact that the simulation is stopped if a carrier set is generated for which all algorithms fails, although it would be theoretically possible to push the limit even further, mitigates this effect. However, the critical carrier set that stops the simulation is still counted in the statistics. This problem does not effect the comparison of the algorithms between themselves, since it is the same for all.

In terms of the evaluation parameters, the successful allocations percentage can be used to compare how often the algorithms fails to allocate a critical set. However, it does not provide any information on how far the algorithms are from succeeding. The average allocated slots parameter gives a better measure on how much benefit these successful

allocations actually gives. When evaluating these values it is important to keep in mind that the absolute value of allocated slots depends on how large the carrier set is. If we look at the varying line bit rates test scenario, the average number of allocated slots is increasing for all algorithms. This cannot be used to evaluate whether the algorithm performs better for higher line bit rates, only how well it performs with respect to the others. The reason for this is that the average carrier set size will increase for higher line bit rates, as the capacity of the basic frame increases. When testing the algorithms on four different line bit rates, there will be at least one link supporting 10.1 Gbps line bit rate, resulting in potentially larger carrier sets. This can explain why the average allocated slots increases for more line bit rates, although the amount of successful allocations indicates that the algorithm performs worse. However, when examining the results for average allocated slots of varying configuration sizes, it generally increases in the beginning, but decline when using more than three REs. With larger configuration sizes, the probability of using higher line bit rates increases, which should increase the average number of allocated slots. On the other hand, lower line bit rates in the configuration may cause the allocation to fail at an early stage, although there is a lot of capacity left in the higher regions. For configurations with many links using low line bit rates, the risk for this increases. We can also take into consideration the results from Section 7.1, where we saw that the fragmentation due to subframe use increases for larger configurations, which could also be a factor.

7.2.2 Algorithm Evaluation

By looking at the individual algorithm simulations, we saw that offline modifications such as sorting and FC IQ-format selection strategies can improve the performance. For HFF and CFF, we saw that only using FC generally had more effect than only using SA. For example, Fig. 6.2.6 showed that the sorting led to small or none improvement on its own when using four different line bit rates. When many different line bit rates are used in a configuration, the risk for fragmentation due to the allocation order decreases for HFF and CFF (which are equivalent in this case), as the risk of having multiple carriers destined for different REs allocated in the same region decreases. The implementation of HFF and CFF can be seen as a "natural" sorting in terms of line bit rates. For the varying REs, the number of line bit rates are randomly generated, and the case of four different line bit rates is not as common. In addition, for the case of using seven REs, there will always be multiple REs sharing the same line bit rate, since the maximum number of line bit rates is less than the number of REs.

When comparing the best online and offline algorithms, it is clear that there is a lot to benefit from using the offline algorithms such as SA/LFF-FC and RB-FC, especially for more complex configurations, such as more REs and different line bit rates.

7.2.3 Fragmentation due to IQ-selection and Allocation Strategy

From the results, it is clear that both the IQ-selection and allocation strategies can contribute to an improved performance. It also showed that the contribution from each of them depends on the configuration. For large configuration sizes, the effect from both of them increased, but especially the IQ-selection got more significant. This reflects the results in Section 6.1.1, where we saw that the benefits of FC compared to MF increases for larger configuration sizes. As for an increased number of line bit rates, the share of contribution from FC to the overall improvement decreased for more than two line bit rates. As discussed earlier, the carrier sets get larger for higher line bit rates. From the

results in Section 6.1.2 we saw that the difference between FC and MF decreases for larger carrier sets, which could explain the development we see here as well.

7.2.4 Improvement Suggestions

The purpose of this thesis was to investigate the possibilities of reducing fragmentation for different IQ-selection and allocation algorithms. For the IQ-selection, a comparison was possible to do with the optimal solution, which means that we can actually give a quite trustworthy prediction on how much room there is left for improvement if using the model that was designed in this thesis. However, this model is quite basic, and does for example only consider single-link connections between REs. We also saw that parameters such as configuration size and carrier set size had impact on the performance. To evaluate more specific cases, it is recommended to look further into different scenarios based on real configurations. In terms of maximum AxC-slots for example, it is likely that higher sizes will be more common in the future and the distribution of AxC-slot alternatives could be reviewed.

For the allocation simulations, there was no proven optimal alternative. This means that it is harder to conclude anything on the maximal improvement potential. We can only say that for the scenarios that were tested in this thesis, it is possible to improve an algorithm at least as much as we have shown. For the purpose of investigating further on potential improvement for the allocation algorithms in this thesis, it is recommended to review the definition of critical carrier sets to be less dependent on the implemented carrier sets, as discussed earlier. It is also recommended to review the theoretical upper bound of carriers in order to avoid invalid carrier sets.

For the purpose of investigating whether the proposed algorithms in this thesis should be implemented or not, it is recommended to evaluate the scenarios for which it would be implemented more specifically, as it was shown that the performance is strongly related to the specific configurations. A possible implementation could be to choose algorithms depending on the configuration parameter values, such as size and line bit rates. However, the most important recommendation before implementing is to do a proper complexity analysis of the alternatives. The complexity has not been considered in this thesis, except for the limitations it has brought on to the simulation possibilities. Nor has the challenges of restructuring the existing system, which is using a first-come-first-serve algorithm, into an offline mode been considered.

We have shown that there is significant room for improvement in terms of interconnect resource allocation within CPRI. However, the results has not been compared to other alternatives, such as eCPRI. Therefore, we cannot decide whether CPRI will be able to manage the transition to 5G or not based on these result. However, as mentioned in Section 1.6, CPRI will be used for many years to come in existing systems. Thus, the findings in this report shows that there is room for improvement, in order to better manage the transition. From an environmental perspective, it is also desirable to increase the durability of the existing systems. It should also be said that the investigation of potential improvement was based on practical examples. Therefore, it is limited to the algorithms developed in this thesis. It is likely that even further improvements of the interconnect resource allocation for CPRI are possible.

In terms of algorithm implementations, it could also be of interest to analyze them internally, to see which part of the algorithms that are the most crucial for their performance. An example of this could be to investigate exactly how much the region-

connection functionality in RB contributes to the overall performance of RB. This would preferably be done in combination with a complexity analysis, in order to weight the utility and complexity against each other for different functionalities.

8

Conclusions

In this thesis, it was investigated how fragmentation due to IQ-format selection and allocation strategy can be reduced in a CPRI transport link. A model of the CPRI frame structure and RBS configuration was designed, as well as a simulation environment, in which both random and deterministic problem instances can be generated. Algorithms for both IQ-format selection and allocation onto a modelled CPRI basic frame can be implemented and evaluated in the simulation framework. It was found that there are potential to improve the performance of CPRI by reducing the fragmentation, both by changing the IQ-format selection strategy and the allocation strategy onto the basic frame.

For the IQ-format selection, two algorithms, MF and FC, were compared to an optimal algorithm with the goal of minimizing the number of used subframes due to IQ-format fragmentation. This was done without the impact of fragmentation due to allocation strategy. While there is still room for improvement for both MF and FC compared to the optimal, it can be concluded that FC performs better or at least as good as MF for all the investigated scenarios. To further investigate the IQ-format selection algorithms it is recommended to specify more specific test scenarios, as well as examining the complexity of the algorithms in detail.

For the allocation algorithms, three online alternatives; HFF, LFF and CFF, and one offline alternative, RB, was implemented. In addition, all of the online alternatives were also implemented with an offline modification, SA. All algorithms were tested with both MF and FC as IQ-format selection strategy.

The results showed that it is possible to increase the performance of CPRI by using offline algorithms. The improvement potential is most significant for larger configurations. As future RBS configurations are expected to become more complex, the benefits of restructuring the CPRI resource allocation are likely to increase.

For the purpose of further investigating the potential improvement for the allocation algorithms, it is recommended to review the definition of critical carrier sets as well as the theoretical upper bound on the carrier set sizes. For the purpose of implementing the proposed algorithms, it is recommended to evaluate more specific scenarios and configurations, as well as the complexity of the different alternatives. The results of this thesis are best used as a motivation for additional studies on more specific implementations and systems. The algorithms has not been tested on actual RBS equipment, but only on the developed model, and the results should be evaluated based on that.

Finally, it can be concluded that although CPRI may not be the best option from a long-term perspective, it will be around for many years to come. In order to better manage

the transition to 5G, it is possible to enhance the performance of CPRI by reducing fragmentation related to IQ-selection and allocation strategies.

References

[1] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, "Things You Should Know About Fronthaul.", *IEEE/OSA Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1077–1083, 2015, ISSN: 07338724.

[2] J. Van Kerrebrouck, L. Breyne, H. Li, J. Bauwelinck, G. Torfs, P. Demeester, and T. Bohn, "Real-time all-digital radio-over-fiber LTE transmission", Nov. 2017, pp. 83–86. DOI: 10.1109/RTUWO.2017.8228510.

[3] A. de la Oliva, J. A. Hernandez, D. Larrabeiti, and A. Azcorra, "An overview of the CPRI specification and its application to C-RAN based LTE scenarios.", *IEEE Communications Magazine*, vol. 54, no. 2, pp. 152–159, 2016. DOI: 10.1109/MCOM.2016.7402275.

[4] Bin Guo, Wei Cao, An Tao, and D. Samardzija, "Cpri compression transport for lte and lte-a signal in c-ran", in *7th International Conference on Communications and Networking in China*, Aug. 2012, pp. 843–849. DOI: 10.1109/ChinaCom.2012.6417602.

[5] E. AB, *Sustainability and Corporate Responsibility Report 2018*, Available at https://www.ericsson.com/495ba6/assets/local/about-ericsson/sustainability-and-corporate-responsibility/documents/2018/sustainability-and-corporate-responsibility-report-2018.pdf (2020/01/23).

[6] M. Nahas, A. Saadani, J.-P. Charles, and Z. El-Bazzal, "Base stations evolution: Toward 4G technology.", *2012 19th International Conference on Telecommunications (ICT)*, pp. 1–6, 2012, ISSN: 9781467307451.

[7] CPRI-Specification, *V7.0 Common Public Radio Interface (CPRI); Interface Specification*, Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation, Alcatel Lucent, and Nokia Networks, 2015.

[8] M. Waqar and A. Kim, "Performance Improvement of Ethernet-Based Fronthaul Bridged Networks in 5G Cloud Radio Access Networks", *Applied Sciences*, no. 14, p. 2823, 2019, ISSN: 2076-3417. DOI: 10.3390/app9142823.

[9] S. Das and M. Ruffini, "A variable rate fronthaul scheme for cloud radio access networks", *Journal of Lightwave Technology*, vol. 37, no. 13, pp. 3153–3165, Jul. 2019, ISSN: 1558-2213. DOI: 10.1109/jlt.2019.2912127. [Online]. Available: http://dx.doi.org/10.1109/JLT.2019.2912127.

[10] J. Bartelt, N. Vucic, D. Camps-Mur, E. Garcia-Villegas, I. Demirkol, A. Fehske, M. Grieger, A. Tzanakaki, J. Gutiérrez, E. Grass, G. Lyberopoulos, and G. Fettweis, "5g transport network requirements for the next generation fronthaul interface", *EURASIP Journal on Wireless Communications and Networking*, no. 89, 2017. DOI: 10.1186/s13638-017-0874-7.

[11] H. Si, B. L. Ng, M. S. Rahman, and J. Zhang, "A novel and efficient vector quantization based cpri compression algorithm", *IEEE Transactions on Vehicular*

Technology, vol. 66, no. 8, pp. 7061–7071, Aug. 2017, ISSN: 1939-9359. DOI: 10.1109/TVT.2017.2670561.

[12] S. H. Kim, H. S. Chung, and S. M. Kim, "Experimental demonstration of cpri data compression based on partial bit sampling for mobile front-haul link in c-ran", in *Optical Fiber Communication Conference*, Optical Society of America, 2016, W1H.5. DOI: 10.1364/OFC.2016.W1H.5. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2016-W1H.5.

[13] P. Zhu, Y. Yoshida, and K.-i. Kitayama, "Fpga-based adaptive space–time compression towards 5g mimo fronthaul", *Optics Communications*, vol. 459, 2020, ISSN: 0030-4018.

[14] C. I, Y. Yuan, J. Huang, S. Ma, C. Cui, and R. Duan, "Rethink fronthaul for soft ran", *IEEE Communications Magazine*, vol. 53, no. 9, pp. 82–88, Sep. 2015, ISSN: 1558-1896. DOI: 10.1109/MCOM.2015.7263350.

[15] L. Li, M. Bi, H. Xin, Y. Zhang, Y. Fu, X. Miao, A. M. Mikaeil, and W. Hu, "Enabling flexible link capacity for ecpri-based fronthaul with load-adaptive quantization resolution", *IEEE Access*, vol. 7, pp. 102 174–102 185, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2930214.

[16] P. Sehier, A. Bouillard, F. Mathieu, and T. Deiss, "Transport network design for fronthaul", in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–5. DOI: 10.1109/VTCFall.2017.8288357.

[17] C. Chang, N. Nikaein, and T. Spyropoulos, "Impact of packetization and scheduling on c-ran fronthaul performance", in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–7. DOI: 10.1109/GLOCOM.2016.7841885.

[18] T. Tashiro, S. Kuwano, J. Terada, T. Kawamura, N. Tanaka, S. Shigematsu, and N. Yoshimoto, "A novel dba scheme for tdm-pon based mobile fronthaul", in *OFC 2014*, Mar. 2014, pp. 1–3. DOI: 10.1364/OFC.2014.Tu3F.3.

[19] C. coorporation, *Industry leaders confirm August release for the new CPRI Specification for 5G*, Available at http://www.cpri.info/press_20170303.html (2020/05/11).

[20] R. Sadykov and F. Vanderbeck, "Bin packing with conflicts: A generic branch-and-price algorithm", *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 244–255, 2013. DOI: 10.1287/ijoc.1120.0499. eprint: https://doi.org/10.1287/ijoc.1120.0499. [Online]. Available: https://doi.org/10.1287/ijoc.1120.0499.

[21] D. Pisinger, "A minimal algorithm for the multiple-choice knapsack problem", *European Journal of Operational Research*, vol. 83, no. 2, pp. 394–410, 1995, EURO Summer Institute Combinatorial Optimization, ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(95)00015-I. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179500015I.

[22] M. M. Baldi, R. Tadei, G. Perboli, and L. De Giovanni, *The Generalized Bin Packing Problem with Bin-Dependent Item Profits*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise ..., 2017.

[23] L. Fortnow, "The Status of the P versus NP Problem", *Commun. ACM*, vol. 52, no. 9, pp. 78–86, Sep. 2009, ISSN: 0001-0782. DOI: 10.1145/1562164.1562186. [Online]. Available: https://doi.org/10.1145/1562164.1562186.

[24] K. Iyer, "Bin packing - An approximation algorithm", Department of Computer Scienceand Engineering, National Institute of Technology Tiruchirapalli, Tech. Rep., 2008.

A

Result Table Values

A.1 Minimizing Subframes Table Values

Table A.1.1: *Table values for varying REs, corresponding to the results in Section 6.1.1.*

Number of REs	Subframe success rate [%]			Used bits success rate [%]			Average subframes			Average used bits		
	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC
1	100.000	100.000	100.000	98.955	100.000	100.000	9.917	9.917	9.917	1171.362	1159.119	1159.119
2	100.000	99.735	99.939	98.243	100.000	99.975	10.130	10.157	10.136	1188.947	1168.057	1168.347
3	100.000	99.455	99.879	97.838	100.000	99.952	10.251	10.307	10.264	1200.878	1174.909	1175.474
4	100.000	99.328	99.771	97.564	100.000	99.949	10.256	10.325	10.280	1200.391	1171.153	1171.753
5	100.000	99.163	99.682	97.438	100.000	99.945	10.246	10.332	10.279	1197.838	1167.150	1167.789
6	100.000	99.177	99.676	97.328	100.000	99.948	10.164	10.249	10.197	1185.990	1154.298	1154.895
7	100.000	99.146	99.644	97.332	100.000	99.947	10.354	10.443	10.391	1207.767	1175.5476	1176.175

Table A.1.2: *Table values for varying carriers per set. Corresponding to results in Section 6.1.2.*

Carriers per set	Subframe success rate [%]			Used bits success rate [%]			Average subframes			Average used bits		
	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC
1	100.000	100.000	100.000	94.387	100.000	100.000	2.139	2.173	2.173	228.438	215.615	215.615
2	100.000	98.790	99.791	96.257	100.000	99.900	3.988	4.025	3.985	442.584	426.018	426.443
3	100.000	99.160	99.748	97.204	100.000	99.947	5.817	5.826	5.791	657.947	639.548	639.888
4	100.000	99.233	99.725	97.148	100.000	99.939	7.625	7.631	7.593	875.983	851.004	851.520
5	100.000	99.247	99.765	97.503	100.000	99.946	9.317	9.337	9.289	1079.314	1052.366	1052.935
6	100.000	99.520	99.769	97.912	100.000	99.972	11.065	11.241	11.213	1314.240	1286.803	1287.169
7	100.000	99.439	99.754	97.825	100.000	99.965	12.878	13.056	13.015	1529.764	1496.488	1497.019
8	100.000	99.517	99.788	98.227	100.000	99.971	14.494	15.152	15.110	1784.769	1753.133	1753.643
9	100.000	99.470	99.794	98.355	100.000	99.964	16.626	16.623	16.569	1961.474	1929.216	1929.904
10	100.000	99.545	99.855	98.420	100.000	99.967	18.200	18.637	18.579	2205.529	2170.682	2171.407
11	100.000	99.539	99.864	98.511	100.000	99.968	20.014	20.298	20.232	2402.900	2367.109	2367.863
12	100.000	99.612	99.870	98.669	100.000	99.973	21.723	21.712	21.656	2578.043	2543.741	2544.433

Table A.1.3: *Table values for varying max AxC-slots per carrier, corresponding to the results in Section 6.1.3.*

Max AxC-slots per carrier	Subframe success rate [%]			Used bits success rate [%]			Average subframes			Average used bits		
	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC	OPT	MF	FC
1	100.000	88.494	96.539	89.004	100.000	98.748	1.933	2.184	2.002	164.197	146.143	147.995
2	100.000	91.705	97.659	90.312	100.000	99.177	2.437	2.657	2.495	241.350	217.968	219.776
3	100.000	94.024	98.373	91.992	100.000	99.438	3.048	3.241	3.098	319.503	293.918	295.579
4	100.000	95.736	98.941	92.632	100.000	99.592	3.606	3.766	3.644	393.359	364.377	365.869
5	100.000	97.080	99.199	93.352	100.000	99.741	4.235	4.362	4.269	471.367	440.030	441.174
6	100.000	97.724	99.321	94.117	100.000	99.813	4.780	4.891	4.812	540.185	508.405	509.355
8	100.000	98.137	99.335	94.808	100.000	99.857	5.576	5.682	5.613	635.757	602.749	603.610
10	100.000	98.650	99.543	95.576	100.000	99.902	6.424	6.512	6.453	740.071	707.327	708.024
15	100.000	98.993	99.620	96.462	100.000	99.929	7.734	7.812	7.763	898.683	866.884	867.504
20	100.000	99.262	99.721	97.257	100.000	99.950	9.544	9.615	9.570	1116.756	1086.126	1086.673
30	100.000	99.452	99.808	98.081	100.000	99.962	12.093	12.159	12.116	1423.866	1396.546	1397.071

A.2 Allocation Strategy Table Values

Table A.2.1: *Successful allocations percentage for varying number of REs.*

Number of REs	**Successful allocations [%]**													
	MF							FC						
	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB
1	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
2	48.936	56.578	14.780	68.920	59.84	56.69	69.158	51.076	60.612	16.738	72.772	63.544	60.582	72.916
3	17.776	29.498	2.842	50.184	27.664	29.024	49.890	30.222	47.458	3.284	67.740	44.296	46.936	67.652
4	13.668	25.934	2.376	52.680	17.584	25.334	53.440	26.598	43.080	3.482	69.768	31.730	42.542	70.230
5	8.078	19.510	2.002	46.218	10.552	19.422	46.704	30.120	46.804	2.724	73.260	32.418	46.510	73.856
6	8.424	16.980	1.436	47.820	10.466	16.612	47.676	26.774	42.162	2.978	72.662	29.302	41.656	72.724
7	13.136	15.154	1.552	50.422	13.418	15.008	50.696	28.106	36.708	1.634	71.348	29.052	36.874	72.176

Table A.2.2: *Successful allocations percentage for varying number of line bit rates.*

Number of linerates	**Successful allocations [%]**													
	MF							FC						
	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB
1	24.27	48.802	24.680	49.242	41.238	48.976	49.216	36.322	72.748	36.988	73.058	60.022	72.982	72.568
2	16.302	29.928	1.982	39.002	22.784	27.306	39.488	43.250	65.856	4.588	74.774	53.004	62.500	74.818
3	11.114	20.974	1.856	49.786	11.048	20.942	50.092	24.260	40.532	2.176	69.186	24.274	40.470	69.216
4	11.948	10.588	1.500	55.560	11.860	10.524	56.104	24.990	26.072	1.474	70.73	25.09	25.900	70.852

Table A.2.3: *Average allocated slots for varying number of REs.*

Number of REs	**Average allocated AxC-slots**													
	MF							FC						
	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB
1	65.210	65.210	64.575	64.575	65.110	65.110	65.233	65.210	65.210	64.575	64.575	65.110	65.110	65.233
2	80.425	80.884	47.303	82.349	81.546	81.329	81.494	80.521	81.090	47.413	82.551	81.731	81.522	81.691
3	79.646	80.870	40.507	83.129	80.528	80.669	82.863	80.463	82.072	40.749	84.446	81.745	82.002	84.113
4	76.481	77.884	37.065	81.367	77.049	77.763	81.575	77.941	80.058	37.368	83.478	78.949	80.039	83.726
5	72.801	74.337	34.549	79.375	73.250	74.434	79.373	74.797	77.066	34.891	81.980	75.398	77.152	82.211
6	70.843	72.283	33.348	77.224	71.031	72.307	77.348	73.028	75.291	33.662	80.191	73.363	75.455	80.410
7	68.550	69.835	32.732	76.123	68.795	70.064	75.888	71.005	84.545	33.103	79.165	71.139	73.157	79.075

Table A.2.4: *Average allocated AxC-slots for varying number of line bit rates.*

Number of linerates	**Average allocated AxC-slots**													
	MF							FC						
	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB	HFF	SA/HFF	LFF	SA/LFF	CFF	SA/CFF	RB
1	58.725	60.192	58.844	60.287	59.539	59.966	60.219	59.511	61.655	59.667	61.776	60.718	61.419	61.723
2	72.347	74.351	42.004	75.547	73.695	74.208	75.647	74.644	77.794	42.483	78.7293	76.481	77.414	78.878
3	75.358	76.772	33.506	81.012	75.586	76.742	81.037	77.146	79.245	33.758	83.287	77.374	79.191	83.322
4	76.765	76.985	30.672	84.159	75.843	76.291	84.197	77.891	88.700	30.747	85.792	77.199	77.904	86.104