INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Graph coloring techniques for planning dynamic optical networks

Pedro Afonso Fernandes Fonseca

Master Degree in Telecommunications and Computer Engineering,

Supervisor:
PhD Luís Gonçalo Lecoq Vences e Costa Cancela, Assistant Professor,
Iscte-IUL

Co-Supervisor:
PhD João Lopes Rebola, Associate Professor,
Iscte-IUL

November, 2021

iscte

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

Graph coloring techniques for planning dynamic optical networks

Pedro Afonso Fernandes Fonseca

Master Degree in Telecommunications and Computer Engineering,

Supervisor:
PhD Luís Gonçalo Lecoq Vences e Costa Cancela, Assistant Professor,
Iscte-IUL

Co-Supervisor:
PhD João Lopes Rebola, Associate Professor,
Iscte-IUL

November, 2021

To my family

Acknowledgment

I want to thank ISCTE-IUL for these five years and all my teachers. Special thanks also to the Instituto de Telecomunicações, for the access given to its resources. I would like to thank the following people for helping me finish the work.

Firstly, I wish to show my appreciation to professors Luís Cancela and João Rebola for their constant follow-up in this work. In the most complicated situations, they always managed to guide me in the right direction. The constant presence of professors throughout the process was essential for completing the work.

I wish to extend my special thanks to my friends who made this journey with me, especially to Sandro Isqueiro, Sofia Pérsio and Marco Silva. Even though they were also doing their dissertations, they always had time to leave positive words for me and were my main listeners.

Finally, I want to thank all my family who have always supported me, especially my mother, father and sister, who often had to deal with my frustrations. Despite being a very private person, they always managed to find solutions to calm me down in the most stressful moments.

Resumo

As redes ópticas dinâmicas serão cruciais nas comunicações ópticas globais nos próximos 5-10 anos. Os principais impulsionadores deste dinamismo são os serviços on-demand, suportados por aplicações como computação em nuvem e computação em grelha, conduzindo à necessidade de uma infraestrutura de rede cada vez mais dinâmica. Ferramentas de planeamento de rede eficientes, que lidam com os problemas de encaminhamento e atribuição de comprimentos de onda serão de extrema relevância neste cenário dinâmico.

Neste trabalho foi desenvolvido um simulador para o planeamento de redes ópticas dinâmicas, e várias redes reais foram testadas, como a National Science Foundation Network, a British Telecom, a US Backbone Network e também redes bidirecionais em anel. Neste simulador, implementa-se um algoritmo de coloração de grafos denominado algoritmo Small-Bucket, que permite a ocorrência de recolorações de nós. Este algoritmo é comparado com o algoritmo First-fit, em termos de probabilidade de bloqueio, número de recolorações, número de cores usadas e tempo de simulação.

Conclui-se que o algoritmo Small-Bucket produz menores probabilidades de bloqueio do que as obtidas com o algoritmo First-fit. No entanto, para atingir essas baixas probabilidades de bloqueio, o algoritmo Small-Bucket faz uso de um maior número de comprimentos de onda e recolorações.

Palavras-chave: Algoritmo Small-Bucket, Coloração de Grafos, Encaminhamento e Atribuição de Comprimentos de Onda, Redes Ópticas Dinâmicas.

Abstract

Dynamic optical networks will be crucial in global optical communications in the next 5-10 years. On-demand services, fuelled by applications such as cloud computing and grid computing, are the main drivers for the availability of an increasingly dynamic network infrastructure. Efficient network planning tools that deal with Routing and Wavelength Assignment problems are of paramount relevance in this dynamic scenario.

In this work, a simulator for planning dynamic optical networks was developed, and several real networks were tested, such as National Science Foundation Network, British Telecom, US Backbone Network, and also bidirectional ring networks. In this simulator, we have implemented a graph coloring wavelength assignment algorithm named Small-Bucket algorithm that allows recoloring to occur. A comparison performance with the First-fit algorithm is performed in terms of the blocking probability, number of recolorings, number of colors used and simulation time.

It is concluded that the Small-Bucket algorithm originates lower blocking probabilities than the ones obtained with the First-fit algorithm. However, to reach these low blocking probabilities, the Small-Bucket algorithm makes use of a larger number of wavelengths and recolorings.

Keywords: Dynamic Optical Networks, Graph Coloring, Routing and Wavelength Assignment, Small-Bucket algorithm.

Contents

List of Figures

List of Tables

List of Acronyms

BT British Telecom
CDC Colorless Directionless Contentionless
EASDC European Aeronautical and Space Defense Company
FAR Fixed-Alternate Routing
GUI Graphical User Interface
IoT Internet of Things
LSP Label Switched Path
NSFNET National Science Foundation Network
OSPF Open Shortest Path First
OSPF-TE Open Shortest Path First - Traffic Engineering
OXC Optical Cross-Connect
PCE Path Computation Element
PLR Packet Loss Ratio
ROADM Reconfigurable Optical Add-Drop Multiplexer
RWA Routing and Wavelength Assignment
SaaS Software-as-a-Service
TCP Transmission Control Protocol
ToRs Top of Racks
UBN US Backbone Network
WA Wavelength Assignment
WDM Wavelength-Division Multiplexing

List of Symbols

A	Traffic intensity offered to a link in the optical network
$A_{network}$	Offered network load
A_{node}	Offered traffic per node
A_{path}	Traffic per path
a_s^l	Traffic per optical path
B	Blocking probability for point-to-point links
b_d	Number of blocked traffic demands
b_l	Blocking probability of a link l
b_{lim}	Maximum number of blocked traffic demands in Monte-Carlo simulation
BP	Blocking probability of a traffic demand
B_s	Blocking probability of the path s
c	Number of parallel circuits and maximum size of the system $M/M/c/c$
C	Maximum number of colors per bucket used in Small-Bucket algorithm
d	Number of levels used in Small-Bucket algorithm
E	Set of edges that form a graph G
E_{Rx}	Relative error of a simulated blocking probability
e_x	Edge number x of the graph G
G	Graph of tested network
g	Adjacency matrix of the graph G
iat	Time between arrivals characterized by a negative exponential variable
M	Markovian process
N_c	Number of circuits/wavelengths per link
N_d	Number of simulated traffic demands
N_{nodes}	Number of network nodes
N_R	Number of vertices during last RESET in Small-Bucket algorithm
p	Number of paths per node
R_s	Set of links in the path s
s	Number of buckets per level used in Small-Bucket algorithm
st	Duration of each traffic demand
$T(GbE)$	Traffic matrix of the graph G in GbE units
$t_{arrivalt}$	Arrival of traffic demand at instant t
T_{sim}	Simulation time, from the arrival of the first demand until the arrival of the last traffic demand
V	Set of vertices that form a graph G

v_y	Vertice number y of the graph G
λ	Arrival rate of traffic demands
λ_x	Assigned wavelength number x
μ	Service rate of traffic demands

CHAPTER 1

Introduction

1.1. Motivation and Background

Nowadays, optical transport networks are usually quasi-static, in the sense that connections often remain in service for a long period of time [1]. Nevertheless, with the need for on-demand services, fuelled by applications such as cloud computing and grid computing, together with the availability of an increasingly dynamic network infrastructure, it is expected that todays' quasi-static optical networks turn into dynamic optical networks in the next 5-10 years [1].

In the optical network layer, routing and wavelength assignment (RWA) are fundamental functions to transport data in an efficient way. In particular, for static optical networks, several wavelength assignment (WA) algorithms have been used along the years, such as the First-fit, Most-used, and Random [1]. These algorithms were initially designed for dynamic networks. Before using these algorithms some kind of sorting strategy must be applied [1]. Graph coloring techniques have also been used for WA in static networks. These techniques aim at coloring the vertices of the network path graph, so that two adjacent vertices have different colors. An example of such algorithms is the Greedy algorithm [1].

Regarding, dynamic optical networks, the heuristics used for WA in static networks can also be used, but in this case, the sorting strategy is not necessary. In a dynamic environment, every time a traffic demand arrives, a new wavelength must be found for routing the respective demand, without changing the wavelengths already in use in the network. This scenario can lead to a high blocking probability, when innumerous almost simultaneous traffic demands arrive, and the network reaches its peak traffic load. In this case of network congestion, the network can be alleviated by letting wavelength reconfigurations to occur [2], in order to reduce the blocking probability.

To the best of our knowledge, graph coloring techniques for WA in optical dynamic networks have not been studied yet. A simple graph coloring WA algorithm for this scenario could be based on running the Greedy algorithm every time a new traffic demand arrives, which implies that all the vertices should be recolored. Another algorithm example is using a technique that just colors the demand that has arrived, which would be equivalent to the First-fit algorithm. In this work, we implement and analyze a graph coloring algorithm for the dynamic scenario that does not requires the recoloring of all the vertices. This algorithm is named Small-Bucket algorithm and was initially proposed in [3]. In the Small-Bucket algorithm, the traffic demands are distributed into a set of buckets, each bucket with its own set of wavelengths (colors). A comparison with the

First-fit algorithm is performed, in terms of the number of wavelengths used, blocking probability and simulation time.

1.2. Objectives

The main objectives of this dissertation are:

(1) Study some of the RWA algorithms for dynamic optical networks. In particular, for the WA problem, graph coloring techniques will be studied, such as the Small-Bucket algorithm [3];

(2) Implementation using the Matlab simulator of the RWA algorithms studied in first step;

(3) Evaluation, through simulation, of the blocking probability of a traffic demand for various networks topologies;

(4) Performance comparison between the Small-Bucket graph coloring algorithm used for the WA problem with the First-fit algorithm, in terms of blocking probability, simulation time, number of recolorings and number colors used.

1.3. Dissertation Organization

This dissertation is organised in five chapters as described in the following:

Chapter 1 presents the background and motivation of this work, as well as the identification of its main objectives. It also shows how this dissertation is organised and the main contributions of this work.

Chapter 2 focuses on the main concepts concerning the area of dynamic optical networks. Network topologies and elements, definition of adjacency and traffic matrices of the network are presented. Some basic concepts regarding the RWA problem are presented and the RWA algorithms studied in this work are described with detail.

Chapter 3 presents the planning tool developed to study dynamic optical networks, resorts to a flowchart for its explanation and shows some examples. The developed Matlab simulator is found in the repository of this dissertation [4]. The routing algorithm studied and implemented in this chapter is the Fixed-Alternate Routing (FAR). The implemented WA algorithms are the First-fit, Most-used, Random and the Greedy graph coloring algorithm, considering different network topologies. After their software implementation, a study is carried out to evaluate the performance of these WA algorithms in terms of the blocking probability and simulation time. The implementation of the several algorithms is validated by comparison with other works [5], [6]. An analytical formalism based on the reduced load approximation is also used to validate our simulator in a simple network scenario.

In Chapter 4, the Small-Bucket algorithm is studied as a graph coloring technique, for WA. After explaining the Small-Bucket algorithm, supported by examples and a flowchart, its performance study is carried out by comparison with the First-fit algorithm, in terms of blocking probability, simulation time, numbers of colors used and number of recolorings. This study is performed for various network topologies, such as US Backbone Network

(UBN) and ring networks with 8 and 16 nodes, and considering several values of the traffic load.

Finally, Chapter 5 summarizes the main conclusions of this work and outlines some proposals for future work.

1.4. Main Contributions

The main contributions of this dissertation are:

- An analytical formalism based on the reduced load approximation was implemented to validate our simulator in a simple network scenario;
- Implementation of two Graph Coloring algorithms, Greedy and Small-Bucket, and also of traditional algorithms, such as First-fit, Most-used and Random, for WA in dynamic optical networks;
- Performance study of the Small-Bucket algorithm in terms of blocking probability of a traffic demand, simulation time, number of colors used and recolorings, for several real network topologies and comparison with the First-fit algorithm;
- The Small-Bucket algorithm leads to better results in terms of blocking probability, assuming $C = N_c$, but require fast tunable transceivers, which can bring out technological problems, besides cost and management issues.

CHAPTER 2

Fundamental Concepts in Dynamic Optical Networks

2.1. Introduction

This chapter covers the basic concepts of dynamic optical networks, including motivations, topologies, architectures and optical planning algorithms. In section 2.2, the motivations for dynamic optical networks are described, and the main applications and architectures are presented. In section 2.3, the most basic concepts of optical network planning will be described, such as the concepts of graph, optical links, traffic demands, optical paths, adjacency matrix and traffic matrix. In the last sections of this chapter, the main objective of this dissertation, RWA, will be discussed. Some of the algorithms will be later implemented and validated in simulation. In section 2.4 it is presented the main routing algorithms in static optical networks, namely Fixed-Routing and Alternative-Path Routing. In section 2.5 it is discusses the main routing algorithms in dynamic optical networks, namely Adaptive Routing and Fixed-Alternate Routing. At the end of the chapter, the WA algorithms are presented. In the WA algorithms, the focus goes to First-fit, Most-used, Random adapted to dynamic networks and to Graph Coloring techniques (Greedy, Small-Bucket and Big-Bucket algorithms).

2.2. Motivation for Dynamic Optical Networks

Graph coloring techniques for network planning of dynamic optical networks have not yet been studied and applied [1]. Most optical networks are considered quasi-static, since the duration of the optical connections, which represent reserved paths to carry traffic demands, generally remain active for a long time period, while in dynamic optical networks, it is expected that this duration would be in the order of a few milliseconds/seconds [1]. According to the work [1], within 5 to 10 years, it is expected that quasi-static optical networks will become dynamic optical networks. This transformation is due to the need to support new technologies, like 5G and cloud services (e.g. video).

Dynamic networks will be a reality in the near future because there are services that require these characteristics and also because there is already technology capable of implementing this networks, such as Colorless Directionless Contentionless (CDC) Reconfigurable Optical Add-Drop Multiplexer (ROADM) and tunable transponders. ROADMs were an important development, avoiding manual interventions on the network nodes. In dynamic optical networks, whose services require more variable optical connections it is necessary to reduce manual intervention in the network to a minimum. It can thus be said that optical transport networks are becoming increasingly configurable, in which the provisioning process is initiated. Thus, these configurable networks take advantage of network elements, such as ROADMs, which can be remotely reconfigured to operate any wavelength and also of adjustable transponders, which can be adjusted to transmit/receive any of the supported wavelength with various modulations formats and bit rates.

2.2.1. Dynamic Optical Network Applications

The growing increase of 5G and cloud services fosters the gradual transition to dynamic networks [1]. Example of cloud services are cloud computing and grid computing applications. Nowadays, cloud computing is performed more closely to the costumers, creating the called Edge Cloud architecture concept that is transforming the telecommunications landscape from a user-to-user or user-to-object, to an object-to-object communication paradigm. In this kind of communications the exchange of data is for a very short time (sub-second), which required a highly dynamic network [7].

In cloud computing, service provider data center resources are used in a large scale for tasks such as application storage, backup, storage, content delivery and web storage [8].

Grid computing corresponds to a computer network, in which the resources of each computer, such as processing, memory and data storage, are shared with other computers in the system. This type of computing is used in companies to improve operational efficiency, increase employee productivity, accelerate business processes, improve redundancy and resilience [8]. One of the application examples is the European Aeronautical and Space Defense Company (EASDC), whose challenge corresponds to the construction of an on-demand computing model for the simulation tools used by engineers, in order to reduce the analysis time. The dynamic network is a very attractive solution, since the time for setting up an optical connection must be in the order of milliseconds to meet human tolerance, in iterative applications.

2.2.2. Dynamic Optical Network Architectures

According to [1], there are four types of dynamic optical network architectures: centralized; distributed; combination of the previous two; Edge Cloud network [7].

In The Edge Cloud network, distributed data centers are important points of the architecture. Edge Cloud emerged as an evolution of the Central Cloud network, due to the long propagation time in the optical fronthaul connection, between antennas and the respective data center. This architecture offers the opportunity to support low latency and multiple traffic demands. With this architecture it was possible to develop new services such as the Internet of Things (IoT), communication between static objects (for example, sensors) and even develop support for 5G applications. With these applications, supported in increasingly dynamic networks, any object can request a service with any other object, to potentially exchange data for a short time. This communication pattern increases the volatility of time dependence on traffic between data centers. New optical network architectures should provide a packet loss ratio (PLR) below than 10^{-10}. Protocols such as the Transmission Control Protocol (TCP) can be used to comply with the ideal PLR. Table 2.1 shows the main requirements of this architecture, whose values are estimates, since the exact values depend on the application. Most optical networks, current and future, do not allow compliance with the requirements of Table 2.1, because they still depend on technologies that use infrastructures unable to support dynamic traffic.

Table 2.1. Future Edge Cloud requirements and characteristics.

End-to-end service turn-up time	$< 1ms$
End-to-end latency (excluding propagation)	$10 - 100\mu s$
End-to-end jitter	$< 1\mu s$
End-to-end Packet loss rate	$<< 10^{-10}$
Number of machines in each edge cloud	200
Typical end-to-end propagation distance	$1 - 50km$

To establish the connection, the originating node uses signaling along the calculated path. An example of a centralized architecture is the Software Defined Networking architecture. The Centralized Architecture is characterized by the existence of a single point in the network, where the routing and resource allocation functions are performed. A very important aspect in this model corresponds to the concept of Path Computation Element (PCE), a computing platform responsible for the routing. In this architecture, a request for an optical path to a destination node is made by the originating node to the PCE and the PCE responds with the necessary path and resources. To establish the connection, the originating node uses signaling along the calculated path. In Distributed Architecture, each node calculates the path to the destination without consulting external entities. In order for the nodes to obtain necessary information for the routing, the nodes must have, among other information, data related to the network topology, disseminating this information through, for example, the OSPF-TE protocol. An example of a distributed architecture is the Generalized Multiprotocol Label Switching architecture

The main advantages of the centralized architecture are: avoiding the containment of resources during the configuration phase of the optical connection, implementing the processing and memory resources in a single element. The disadvantages of the centralized architecture correspond to the fact that the PCE can be overwhelmed with new requests and the high security risk. From the point of view of distributed architecture, the main advantage is the very fast configuration time and the disadvantages correspond to the need for predominant processing resources, the loss of optimization in the calculation of paths and the amount of resource contention.

2.3. Basic Concepts of Optical Network Planning

The planning of optical networks involves different problems such as the definition of the location of the nodes and the design of the physical topology, the routing of traffic and dimensioning the capacity of the connections, the wavelength assignment, the aggregation of traffic and, also, the protection and restoration. An optical network is commonly described by a graph $G = (V,E)$. Each undirected edge represents a pair of point-to-point unidirectional fiber optical connections, connecting a pair of nodes. The set V

represents the set of vertices or nodes of the network (switches, routers or ROADMs), while the set E represents the set of edges of the network (optical fibers). The graph on the right side of Figure 2.1 shows a simple example of the representation of the optical network in a graph.

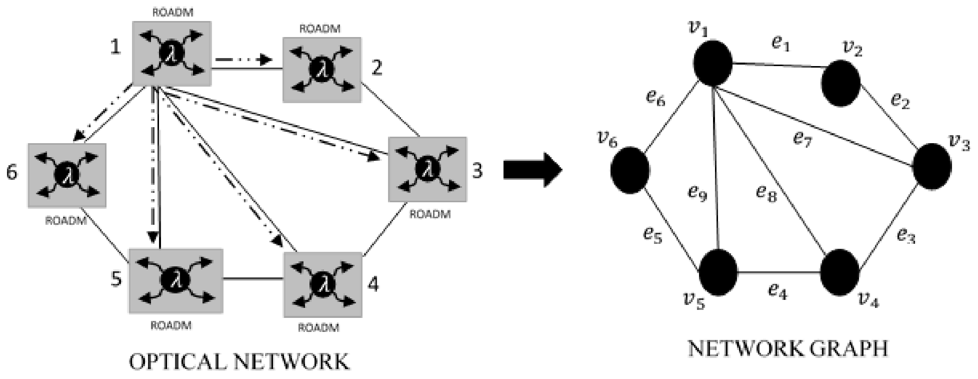


FIGURE 2.1. Graph of the Network - representation of the network.

The terminology in the context of the planning of optical networks should also be defined. It is important to describe the concept of optical link, which means a physical connection between two adjacent nodes. As example in Fig. 2.2, a possible optical link between two nodes is e_7, which links the vertices v_1 and v_3. One of the main concepts in the process of RWA corresponds to the concept of optical path, which is a sequence of links between a source node (s) and a termination node (t).

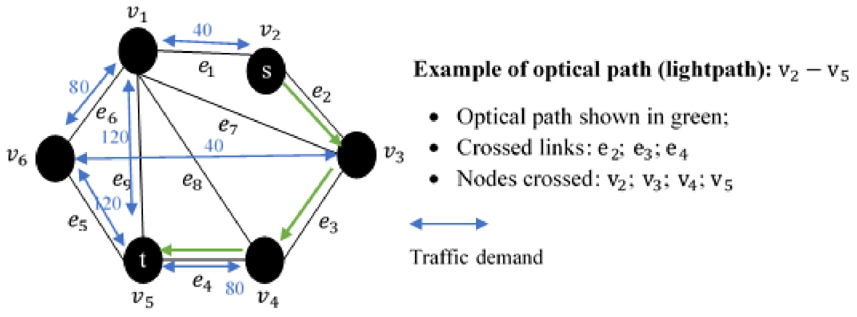

v_1
v_2
v_3
v_4
v_5
v_6
s
t
e_1
e_2
e_3
e_4
e_5
e_6
e_7
e_8
e_9
40
80
120
40
120
80
Example of optical path (lightpath): $v_2 - v_5$
Optical path shown in green;
Crossed links: e_2; e_3; e_4
Nodes crossed: v_2; v_3; v_4; v_5
Traffic demand

FIGURE 2.2. Example of optical paths.

The set of services carried by the optical network is called traffic and an individual traffic demand between two nodes is represented in the form of a traffic demand, corresponding to a logical connection. This traffic demand gives rise to the reservation of an optical path, called a connection. These services carried on the optical network are conditioned by the type of physical topology of the network, which can be of different types, e.g. mesh; ring; star; bus; tree. The network physical topology, allows specifying the interconnection strategy between the nodes. Another way to represent the physical topology of an optical network is through the adjacency matrix. Considering N, number of nodes in a network, the dimension of the adjacency matrix (**g**) is $N{\times}N$. The adjacency matrix specifies the links between every node of the network. If there is a link between i and j, then g_{ij}=1, otherwise, g_{ij}=0. Taking into account the graph of Figure 2.1, the adjacency matrix corresponds to the following matrix

$$g = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.1}$$

While the physical topology specificates the interconnection strategy between the nodes, the logical topology allows to specify how the information flows through the network. Traffic can be described by the number of traffic demands or by the logical connections. Taking as an example, the network described in Fig. 2.1, the blue arrows of the graph of Fig. 2.2 can be used as a logical topology. The logical topology of an optical network can be represented by a matrix, called the demand matrix. The dimension of the demand matrix (d) is $N{\times}N$. The demand matrix allows to specify whether there is a traffic demand between two nodes, i and j. If there is a traffic demand between i and j, then d_{ij}=1, otherwise, d_{ij}=0. Taking into account the graph of Fig. 2.2, the corresponding demand matrix is given by

$$d = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{2.2}$$

It is also possible describe the volume of traffic over a period of time between all nodes in the topology, using a traffic matrix. In transport networks, traffic units correspond to the various types of client signals, such as E3, STM-1, 10 GbE, 100 GbE and Fiber Channel (FC) which must be converted into appropriate traffic units to be used in the transport network. By reusing the network graph, in Figure 2.1, it is possible to describe the volume of traffic between the network nodes in traffic units as represented in Figure 2.2. Taking into account Figure 2.2, the traffic matrix, which describes the volume of traffic, in GbE units, over a period of time between all nodes, can be represented as

$$T(GbE) = \begin{bmatrix} 0 & 40 & 0 & 0 & 120 & 80 \\ 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 40 \\ 0 & 0 & 0 & 0 & 80 & 0 \\ 120 & 0 & 0 & 80 & 0 & 120 \\ 80 & 0 & 40 & 0 & 120 & 0 \end{bmatrix}. \tag{2.3}$$

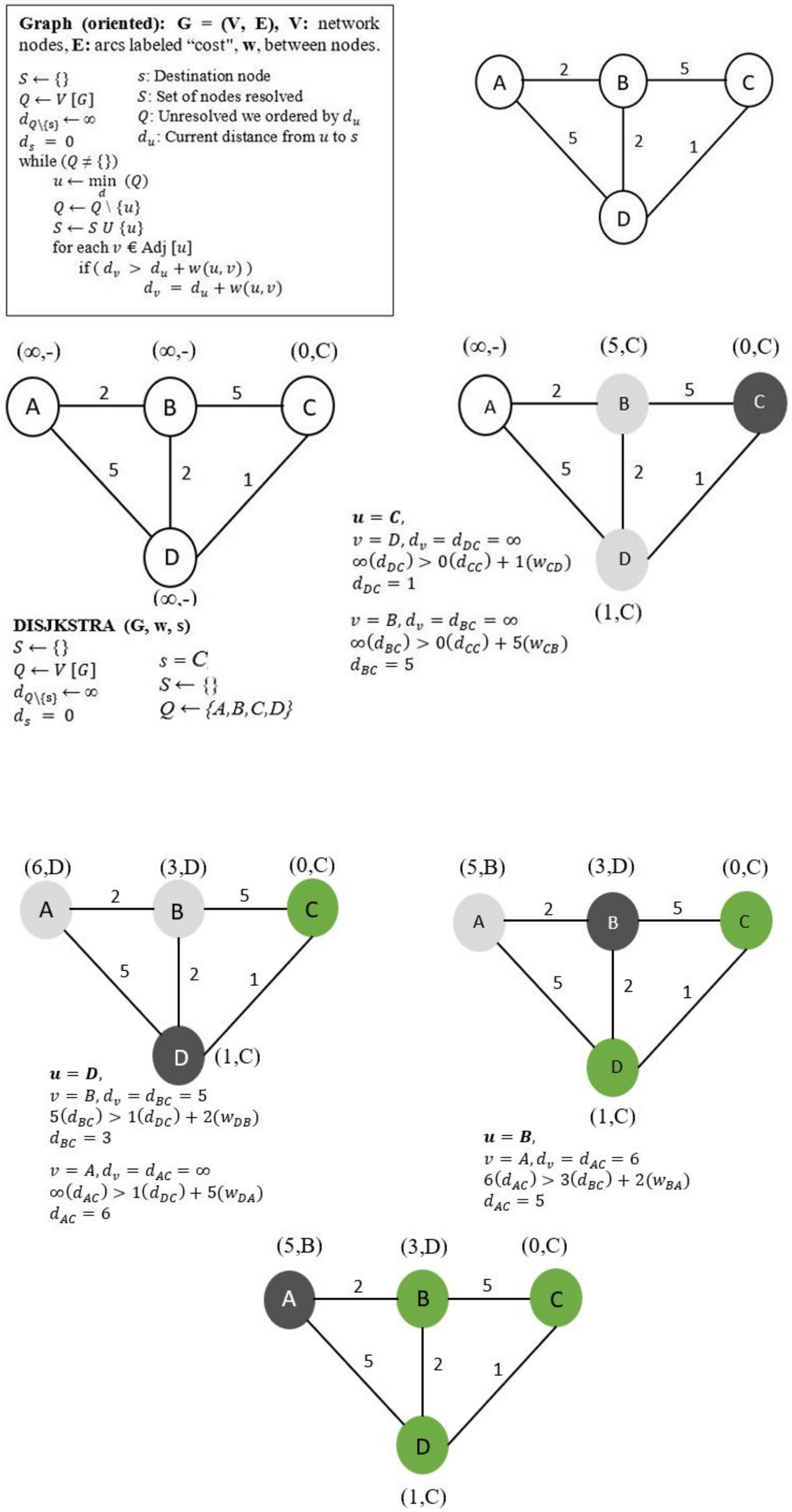
2.4. Routing in Static Optical Networks

The RWA problem consists of two sub-problems: routing and wavelengths assignment. The complexity of this problem is due to the calculation of the path for a traffic demand, continuity and distinct wavelength properties. The same wavelength must be assigned to all connections that form the optical path. Two optical paths must not have the same wavelength on the same link. For static optical networks, RWA algorithms are characterized by: calculation of physical paths for each pair of nodes; organization of the various paths previously calculated in a list; sorting all wavelengths in the list; search for the ideal path and wavelength for the selected optical path.

The optical paths can be determinated using both centralized and distributed architecture. In both cases, the routing algorithms that are used to calculate the paths can be: Fixed-Routing (e.g. Dijkstra) and Alternative-Path Routing (e.g. Yens k-shortest) [1]. Exists several network metrics to evaluate the cost of a connection, such as the path length, delay, delay variation and loss probability. Most static or dynamic routing algorithms generally optimize a link metric, using some variant of the shortest path algorithms, such as Dijkstra's algorithm. Routing algorithms can be classified as online/offline, taking into account the way in which the paths are calculated. In situations where paths are calculated before the optical network goes into operation, the routing algorithm can be classified as offline. In the other scenario, in which the paths are calculated after the network goes into operation, the routing algorithm can be classified as online.

2.4.1. Fixed-Routing

In this type of routing algorithm, a single optical path is calculated for a given traffic demand. An example of a Fixed-Routing algorithm corresponds to the Dijkstra algorithm. Dijkstra allows obtaining the shortest path in a directed or non-directed graph with non-negative weight edges. Dijkstra's algorithm analyzes the entire graph in order to find the shortest path between the origin node and all other nodes. The shortest distance currently known from each node to the origin node is tracked and this value is updated if a shorter path is determined. When calculating the shortest path between the originating node and another node, that node is marked as "visited" and added to the set S (resolved nodes). This procedure is repeated until all nodes in the graph have been added to set S. An example of the application of Dijkstra's algorithm to find shorter paths in a graph (directed or not) without negative weight edges is shown in the Figure 2.3, considering node C as the destination node.



Graph (oriented): G = (V, E), V: network nodes, **E:** arcs labeled "cost", **w**, between nodes.

```
$S \leftarrow \{\}$                  $s$: Destination node
$Q \leftarrow V[G]$                  $S$: Set of nodes resolved
$d_{Q\setminus\{s\}} \leftarrow \infty$          $Q$: Unresolved we ordered by $d_u$
$d_s = 0$                        $d_u$: Current distance from $u$ to $s$
while ($Q \neq \{\}$)
    $u \leftarrow \min_d (Q)$
    $Q \leftarrow Q \setminus \{u\}$
    $S \leftarrow S \, U \, \{u\}$
    for each $v$ € Adj [$u$]
        if ( $d_v > d_u + w(u,v)$ )
            $d_v = d_u + w(u,v)$
```

DISJKSTRA (G, w, s)

```
$S \leftarrow \{\}$
$Q \leftarrow V[G]$             $s = C$
$d_{Q\setminus\{s\}} \leftarrow \infty$     $S \leftarrow \{\}$
$d_s = 0$                   $Q \leftarrow \{A,B,C,D\}$
```

$\boldsymbol{u = C}$,
$v = D, d_v = d_{DC} = \infty$
$\infty(d_{DC}) > 0(d_{CC}) + 1(w_{CD})$
$d_{DC} = 1$

$v = B, d_v = d_{BC} = \infty$
$\infty(d_{BC}) > 0(d_{CC}) + 5(w_{CB})$
$d_{BC} = 5$

$\boldsymbol{u = D}$,
$v = B, d_v = d_{BC} = 5$
$5(d_{BC}) > 1(d_{DC}) + 2(w_{DB})$
$d_{BC} = 3$

$v = A, d_v = d_{AC} = \infty$
$\infty(d_{AC}) > 1(d_{DC}) + 5(w_{DA})$
$d_{AC} = 6$

$\boldsymbol{u = B}$,
$v = A, d_v = d_{AC} = 6$
$6(d_{AC}) > 3(d_{BC}) + 2(w_{BA})$
$d_{AC} = 5$

FIGURE 2.3. Application of Dijkstra's algorithm.

2.4.2. Alternative-Path Routing

In this type of routing algorithm, several alternative optical paths are calculated for a traffic demand. An example of a Alternative-Path routing algorithm corresponds to the Yens K-Shortest algorithm. The Yens K-Shortest Path algorithm is identical to the Shortest Path algorithm, but with the difference that K shortest paths are calculated. It uses the Dijkstra algorithm to calculate the shortest paths. The Dijkstra algorithm is applied initially to find the optical path with the lowest cost and, subsequently, to each removal of optical connections between each pair of nodes that form the initial optical path. Then, an application example, based on the demonstrations of [1] will be presented.

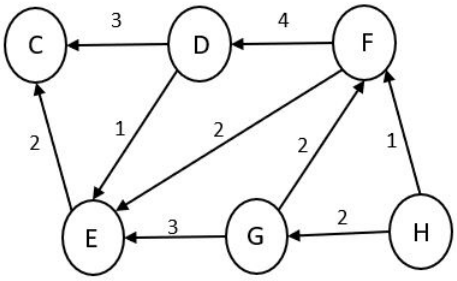


FIGURE 2.4. Yens K-Shortest Path - example of application.

In this example, the Yens K-Shortest Path algorithm is applied to obtain the 2 shortest paths between H and C. The steps followed are:

- apply Dijkstra's algorithm to obtain the lowest cost path between H and C;
- 1^{st} optical path calculated: *HFEC* (Cost = 5);
- remove the *HF* optical link and apply Dijkstra's algorithm;
- 2^{nd} optical path calculated: *HGEC* (Cost = 7).

This type of routing is based on the existence of alternative paths between two nodes. Before adding a traffic demand to the network, a set of K possible paths for that traffic demand is generated. After the arrival of a traffic demand, with a specific origin/destination, one of the paths of the set previously selected is selected to be used.

2.5. Routing in Dynamic Optical Networks

For dynamic environments, whose optical connections are started/finished quite frequently, it would be expected that the routing algorithms of the Adaptative Routing type would be the most efficient, but the latency produced due to the constant need to execute the optical path calculation algorithm takes researchers to look for solutions adapted from Alternative-Path Routing algorithms, to perform routing in dynamic optical networks.

2.5.1. Adaptative Routing

In this type of routing algorithm, the paths between the nodes of the network are calculated after the network starts operating. Taking into account the volatile nature of dynamic optical networks, one would expect that this would be the best method of calculating the path, however due to the high delay produced, due to the need to execute the optical path calculation algorithm for multiple traffic demands, this solution is not very effective in planning, whose main objective is to reduce the latency of services. In conjunction with the current state of the optical network, several metrics can be considered to calculate optical paths, such as cost, traffic on the most loaded connection, the number of hops, the distance of the path, the protection and the signal-to-noise ratio of the path.

2.5.2. Fixed-Alternate Routing

Fixed-Alternate Routing (FAR) strategies can be adapted for dynamic optical networks: before the network goes into operation, the shortest optical paths between any pair of nodes will be calculated. When assigning optical paths for traffic demands, the optical paths determined before the network goes into operation, which are in the best conditions, are used. That is, instead of calculating the shortest path each time a traffic demand arrives, the paths are already pre-calculated and the optical path in better conditions is used.

2.6. WA in Static Optical Networks

The general objective of RWA in static networks is to transport traffic using the smallest number of wavelengths. Each traffic demand must have an optical path (R) and wavelength (WA), in order to be transported in the optical network. In static optical networks, in the process of WA, traffic demands must first be ordered, according to methods such as shortest path first, longest path first or random, and subsequently is applied an algorithm (e.g. First-fit, Most-used or Random) or graph coloring techniques for WA. A very important aspect common to all WA algorithms is the fact that two optical paths with common optical links cannot have the same assigned wavelength.

2.6.1. First-fit Algorithm

In this WA algorithm, all available wavelengths are indexed and the available wavelength with the lowest index is chosen. The algorithms for indexing wavelengths can be, for example, shortest path first, whose traffic demands with the lowest cost on the path appear first in the list, longest path first, whose traffic demands with the highest cost appear first in the list or random ordering, where traffic demands are sorted randomly.

2.6.2. Most-used Algorithm

In this case, the sorting algorithms used to index the wavelengths are the same than in First-fit. In this algorithm, there is a variable that stores the number of times that each wavelength is used per link. To the wavelength that is assigned to more links is given a higher priority and is assigned to the new path if the two conditions are met (continuity of wavelength and distinct wavelengths). If these conditions are not met, the next wavelength on the sorting list with more links is tried.

2.6.3. Random Algorithm

A wavelength is randomly chosen from among all available wavelengths, and it is generally assumed that all wavelengths have equal probability of being chosen. A chain network with 7 nodes (0 to 6) is considered with the WA shown in Figure 2.5. Unshaded regions indicate that wavelength is available on these links.

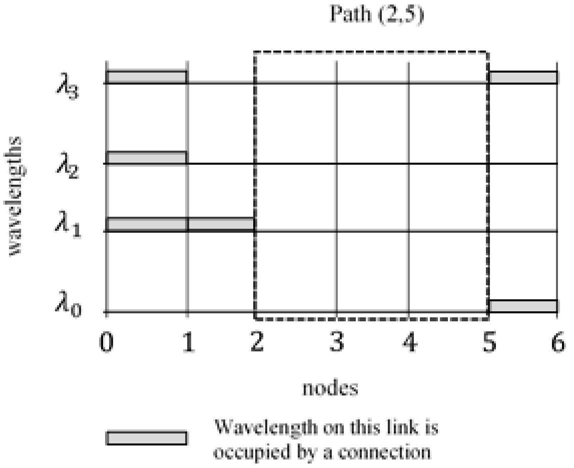


Figure 2.5. First-fit, Most-used and Random application example.

To establish a path between 2 and 5, there are 4 free wavelengths (λ_0 to λ_3). The assigned wavelengths, according to each algorithm, follow the ideas previously described.

Table 2.2. Results of the WA algorithms - Static Optical Network.

Algorithm	Chosen wavelength
First-fit	λ_0
Most-used	λ_1 or λ_3
Random	λ_0,λ_1,λ_2 or λ_3

2.6.4. Graph Coloring Algorithms

Graph coloring algorithms corresponds to a conventional mathematical problem that consists in coloring all the nodes in a graph, so that there are no two adjacent nodes with the same color [1]. An important point of this technique consists of passing the graph of the physical topology of the network *G(V,E)* to a graph *G(W,P)*, whose nodes/vertices (W) are the optical paths and P represents the set of optical links between these nodes.

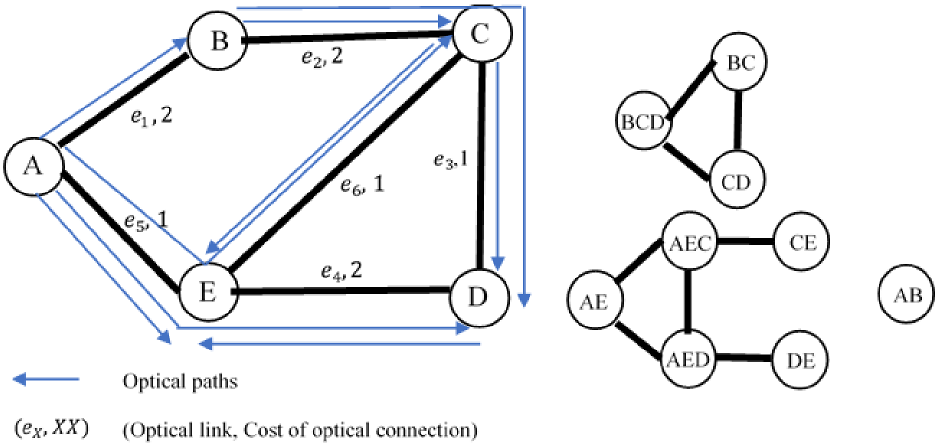


Figure 2.6. Graph of the physical topology (left) and of the optical paths (right).

Taking into account the graphs represented in Figure 2.6, it can be written that the set V is [A,B,C,D,E]; the set E given by [e_1,e_2,e_3,e_4,e_5,e_6]; the set W given by [*AE, AEC, AED, CE, DE, AB, BCD, CD, BC*] and, finally P, the set of connections between these nodes. The procedure to be carried out corresponds to coloring the vertices of the new graph, which are obtained through the following procedure: 2 vertices have a link between them if 2 nodes of each path are coincident. After constructing the graph of the optical paths *G(W,P)*, the ordering strategy is chosen for the coloring of the vertices. The most used strategy for coloring is the Greedy algorithm. In Greedy algorithm, the vertices are sorted by the vertex order (e.g. descending, ascending, random) and then each vertex is colored in such a way that adjacent vertices can not have the same color.

The most common sorting strategy is the descending order strategy. This strategy will be used to exemplify the graph coloring technique. Taking into account the graph in Figure 2.6, the Table 2.3 can be constructed which, for each optical path, lists the number of edges incident at the vertex (i.e., the vertex order) and the assigned wavelength. This strategy consist in sorting the vertices of the graph *G(W,P)*, starting from the first vertex until the last vertice in order (ascending/descending), assigning them with the lowest number color (wavelength) not yet used for a neighbor. The Greedy strategy is more suitable for static optical networks, because it assumes that all traffic demands arrive at the same time, being necessary to order them and then order the optical paths according to the degree of each path. However, it can also be adapted for dynamic networks, as at each arrival of a traffic demand, the Greedy algorithm, previously described, can be executed without sorting by vertex order. In order to make the process of WA in dynamic optical networks efficient, the best way to reserve a wavelength in each of the links that form a path corresponds to assigning only the optical path, generated at the moment, a wavelength adapted to the conditions of the network, without the need to carry out the entire WA calculation for all previous traffic demands.

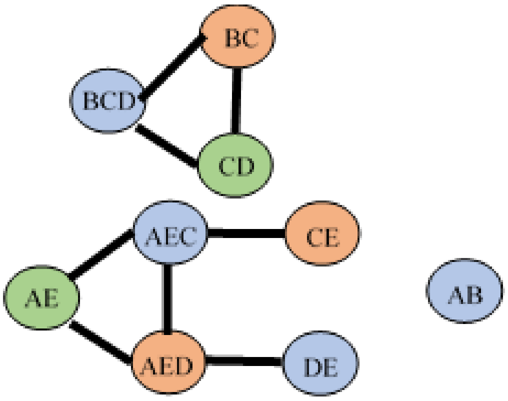


FIGURE 2.7. Coloring of graph G(W, P).

TABLE 2.3. Distribution of wavelengths - Greedy algorithm.

Path	Degree	Wavelength
AEC	3	Blue, λ_1
AED	3	Orange, λ_2
AE	2	Green, λ_3
BCD	2	Blue, λ_1
BC	1	Orange, λ_2
CD	1	Green, λ_3
CE	1	Orange, λ_2
DE	1	Blue, λ_1
AB	0	Blue, λ_1

2.7. WA in Dynamic Optical Networks

In [9], two categories of wavelength assignment in dynamic networks are studied: Online Graph Coloring and Dynamic Graph Coloring.

2.7.1. Online Graph Coloring

Online Graph Coloring considers that over time, more vertices arrive at the graph $G(W, P)$ (calculated optical paths) and with each vertex insertion, the respective coloring of these vertex occurs, without the need to recolor all the vertices of the graph. The goal is to minimize the blocking probability.

One of the Online Graph Coloring algorithms corresponds to the Greedy graph coloring strategy, for dynamic optical networks. This strategy, taking into account the dynamism of the network, considers that the vertices are added and colored over time. The Greedy strategy, previously studied in static environments, considers that all traffic demands arrive at the same time and that all the vertices of optical paths are colored at the same time. In dynamic environments, the strategy is executed to every traffic demand arrival, allowing the coloring of just the new optical path vertex.

2.7.2. First-fit, Most-used and Random Algorithms

The WA algorithms studied in section 2.6 show better results for static networks, whose traffic patterns and optical connections do not vary over time. In dynamic networks, algorithms such as First-fit, Most-used or Random, can be used for the WA. Different methodologies can be considered to assign wavelengths, one that allows to minimize blocking but that increases the complexity of the solution and another that allows to minimize the complexity of the solution, but that makes use of a larger number of wavelengths. To minimize the number of used wavelengths and increase the complexity of the solution, a correct strategy is to execute one of the static environment WA algorithms for all traffic demands, after each update of the optical network. On the other hand, another strategy corresponds to adapt the WA to the current network conditions (considered in the simulator developed), that is, with each update in the network, no WA algorithm is performed, but taking into account the algorithm strategy static (First-fit, Most-used or Random) assign a wavelength to the incoming traffic demand, or otherwise assign a wavelength to a traffic demand that no longer exists. This solution, minimizes the complexity of the solution, but can provide a high number of used wavelengths [9]. In order to demonstrate the operation of each WA algorithm for dynamic optical networks, an example is presented below, with 10 traffic demands arriving at different times. In this example, the strategy considered corresponds to assigning a wavelength for each calculated optical path, using the First-fit, Most-used and Random algorithms.

Example: In the example considered, the network described by the graph in Figure 2.8 will be used. At various time instants, traffic demand arrivals and respective optical routing are simulated, using the number of hops as a metric. At each time point, in addition to the optical routing, the First-fit, Most-used and Random algorithms are executed in order to assign wavelength to the determined path. The maximum number of wavelengths available for assignment is 5. The results of the optical routing and WA algorithms for the exemplified optical network are shown in Table 2.4.

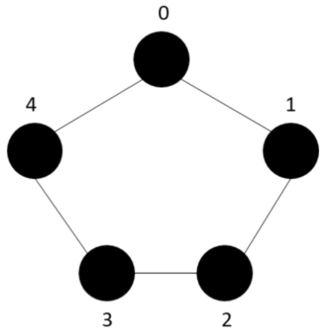


FIGURE 2.8. Ring network graph with 5 nodes used to exemplify First-fit, Most-used and Random in dynamic optical networks.

Table 2.4. Distribution of wavelengths - Traditional algorithms.

Instant	Path	WA - First-fit	WA - Most-used	WA - Random
1	$0-1$	λ_1	λ_1	random(λ_1,λ_2,λ_3,λ_4,λ_5)=λ_3
2	$0-1-2$	λ_2	λ_2	random(λ_1,λ_2,λ_4,λ_5)=λ_4
3	$0-4-3$	λ_1	λ_2	random(λ_1,λ_2,λ_3,λ_4,λ_5)=λ_2
4	$0-4$	λ_2	λ_1	random(λ_1,λ_3,λ_4,λ_5)=λ_1
5	$1-2$	λ_1	λ_1	random(λ_1,λ_2,λ_3,λ_5)=λ_5
6	$1-2-3$	λ_3	λ_3	random(λ_1,λ_2,λ_3)=λ_2
7	$1-0-4$	λ_3	λ_3	random(λ_5)=λ_5
8	$2-3$	λ_1	λ_2	random(λ_1,λ_3,λ_4,λ_5)=λ_1
9	$2-3-4$	λ_2	λ_1	random(λ_3,λ_4,λ_5)=λ_3
10	$3-4$	λ_3	λ_3	random(λ_1,λ_4,λ_5)=λ_1

From the results, it can be concluded that the First-fit and Most-used algorithms produce the same number of assigned wavelengths (3) to satisfy the 10 traffic demands. The Random algorithm, due to its random choice strategy, needs more wavelengths.

2.7.3. Dynamic Graph Coloring

In this algorithms, vertices and edges may appear/disappear over time. The objective of graph coloring techniques adapted to dynamic networks is to maintain an WA solution in a scenario where there are demands/optical connections that require an update of the network over time. A simplistic way is to run the best static algorithm after each new demand [9]. On the other hand, a dynamic graph coloring algorithm allows to maintain an intelligent data structure for the underlying problem, resulting in a lower update time for the solution, compared to the static algorithm. Currently, there are not many studies carried out on graph coloring in dynamic optical networks.

In [3], a study of the number of vertex recolorings that an algorithm needs to maintain a suitable color of a graph under insertion and extraction of vertices/nodes and edges is performed. A recoloring algorithm is an algorithm that maintains a suitable graph coloring, even when that graph undergoes a series of updates. In algorithms of [3], it is assumed that the exclusion of a vertex or connection never invalidates the coloring of the graph and, therefore, the coloring techniques studied in [3] do not perform any recoloring when vertices or connections are ended. In the considered algorithms, only the insertions of vertices (new connections) require resorting.

The central idea of the algorithms presented in [3] corresponds to the distribution of the vertices in a set of buckets, each with its own set of colors used for the coloring of the vertices. The algorithms proposed in [3] are called Small-Bucket and Big-Bucket and differ in terms of the number of buckets and in the size of each bucket. As a rule, there is a sequence of buckets of increasing size, organized by levels and a RESET bucket. The size of each bucket varies with the execution time of the algorithm. Initially, the size depends on the number of vertices in the input graph and, later, it depends on the insertions and exclusions of vertices. The general idea of the algorithms is that when certain buckets are full, the system is redefined to ensure acceptance of new vertices. Redefining the system involves emptying all the buckets in RESET bucket, calculating a new color suitable for every vertex and recalculating the bucket sizes in terms of the current number of vertices.

2.8. Conclusions

The planning of dynamic optical networks appears to support new future technologies, such as 5G applications and cloud services. There is already technology in the market capable of implementing this type of networks, for example CDC ROADMs and tunable transponders. In these networks, whose services require more variable optical connections, it is necessary to minimize manual intervention in the network. Optical transport networks are becoming increasingly configurable, in which the provisioning process is initiated. Thus, these configurable networks take advantage of existing technology.

The planning of dynamic optical networks is not something very discussed in articles in the specialty, however there are some studies on routing and wavelength assignment algorithms. In this chapter, several traffic routing algorithms were presented, both for static and dynamic networks. In static optical networks there are two main types of routing algorithms: Fixed-Routing; Alternative-Path Routing. For dynamic optical networks, routing algorithms can also be classified into two large groups: Adaptive Routing; Fixed-Alternate Routing. Regarding the WA, several algorithms were also studied, namely the First-fit, Most-used, Random, Greedy graph coloring, Small-Bucket and Big-Bucket algorithms.

In the next chapters, some of the algorithms covered in this chapter will be implemented and studied, in terms of the blocking probability of a traffic demand, simulation time, number of colors/wavelengths used and recolorings.

CHAPTER 3

Implementation of a RWA Planning Tool for Dynamic Traffic

3.1. Introduction

In this chapter, the Matlab simulator development is reported, which allows the implementation of some of the RWA algorithms for dynamic optical networks. The developed simulation tool to evaluate the network blocking probability considers various network topologies and traffic patterns and, also allows comparing the performance between different algorithms. Before evaluating the blocking probability, this metric is evaluated in section 3.2, in a single point-to-point link modelled as a $M/M/c/c$ system. The results will be compared with the well-known Erlang-B formula, for validation purpose. Then, in section 3.3 the simulation model for a network with dynamic traffic is explained and the simulator is validated for various network topologies and RWA algorithms, [5] and [6].

3.2. $M/M/c/c$ System

According to Kendall 's notation [10], $A/S/c/K/Q$, A indicates the time distribution between arrivals, S the service time distribution, c the number of parallel circuits, $K(\geq c)$ the maximum size of the system and Q the queue discipline. The point-to-point links in the traditional telephone network with circuit switching is an application example that can be modelled by a $M/M/c/c$ system, where the time distribution between arrivals and the service time distributions follows a Markovian process. In this scenario when all circuits are occupied and a demand arrives, that demand is blocked and the call is lost. In this networks, traffic demands arrive according to a Poisson process with arrival rate λ and service times exponentially distributed with parameter μ. The blocking probability for this point-to-point links can be calculated with the Erlang-B formula [11], given by

$$B = E_{1,N_c}(A) = (A^{N_c}/N_c!)/(\Sigma_{n=0}^{N_c} A^n/n!), \tag{3.1}$$

where $A = \lambda/\mu$ corresponds to the intensity of traffic offered to the system expressed in Erlang and N_c corresponds to the number of circuits.

3.2.1. Simulation Model for a $M/M/c/c$ System

The simulation of the $M/M/c/c$ system is done through an event-based simulation where traffic demand arrivals are simulated following

$$t_{arrival_{i+1}} = t_{arrival_i} + iat, \tag{3.2}$$

where iat is the time between arrivals characterized by a negative exponential random variable

$$iat = -1/\lambda \times ln(u), \tag{3.3}$$

where u is a uniform random variable between 0 and 1. The duration of each traffic demand is simulated through the service time variable

$$st = -1/\mu \times ln(u). \tag{3.4}$$

The time instant corresponding to the end of the traffic demand i is estimated using

$$t_{end_i} = t_{arrival_i} + st_i. \tag{3.5}$$

In Fig. 3.1, an example of the $M/M/1/1$ system is presented, and 13 events are shown.

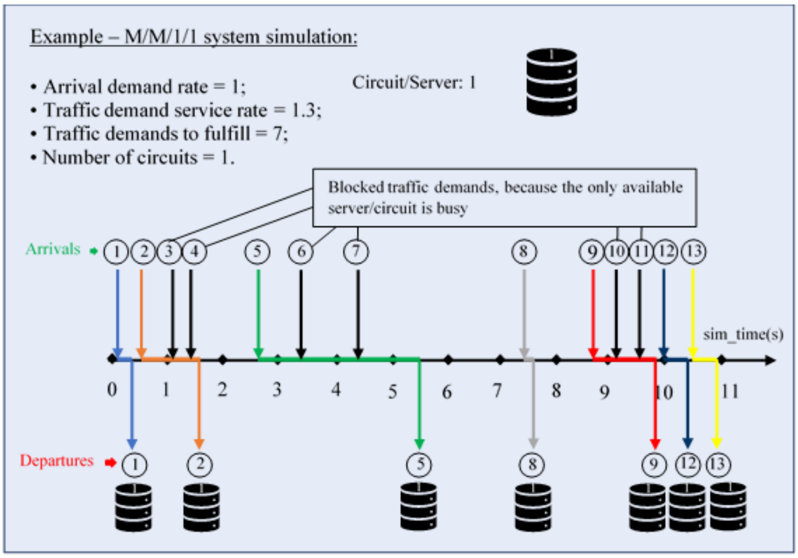

Example – M/M/1/1 system simulation:
• Arrival demand rate = 1;
• Traffic demand service rate = 1.3;
• Traffic demands to fulfill = 7;
• Number of circuits = 1.
Circuit/Server: 1
Blocked traffic demands, because the only available server/circuit is busy
Arrivals
1 2 3 4 5 6 7 8 9 10 11 12 13
sim_time(s)
0 1 2 3 4 5 6 7 8 9 10 11
Departures
1 2 5 8 9 12 13

FIGURE 3.1. Example of a $M/M/1/1$ system simulation.

In this scenario, with one circuit, the system has the capacity to serve only one traffic demand. Situations such as the arrival of traffic demands number 3 and 4, cause blocking of demands, because the circuit/server of the system is busy attending other traffic demand. This situation occurs with 4 more traffic demands. The simulated blocking probability is calculated by dividing the number of blocked traffic demands by the number of arrived traffic demands. The number of blocked traffic demands is 6 (traffic demands 3, 4, 6, 7, 10 and 11). In the example given in Fig. 3.1, the blocking probability estimated by simulation is 6/13.

The flowchart of the simulator that allowed to obtain the results of Figure 3.1, is described in Figure 3.2. The simulator starts by defining the physical topology of the network and calculating the arrival time of the first traffic demand. The stopping criterion is the number of simulated traffic demands, up to 100000 traffic demands. When the maximum number of traffic demands is reached, the simulator calculates the blocking probability of the simulation and the simulation ends. While the maximum number of traffic demands is not reached, two possible events are generated:

- traffic demand arrival (temporal instant of next arrival < temporal instant of demand departure): if the simulator's only circuit/server is busy with an traffic demand, it means that the traffic demand i is blocked and the number of blocked traffic demands is increased. If the circuit is not busy, it means that the traffic demand i is assigned the system circuit, the number of traffic demands arrivals is increased and it is calculated the time of the next demand arrival ($t_{arrival_{i+1}}$) and the time of departure of the traffic demand i (t_{end_i}).

- traffic demand departure (temporal instant of next arrival > temporal instant of demand departure): the simulator increases the number of traffic demands served and decreases the number of traffic demands being executed.

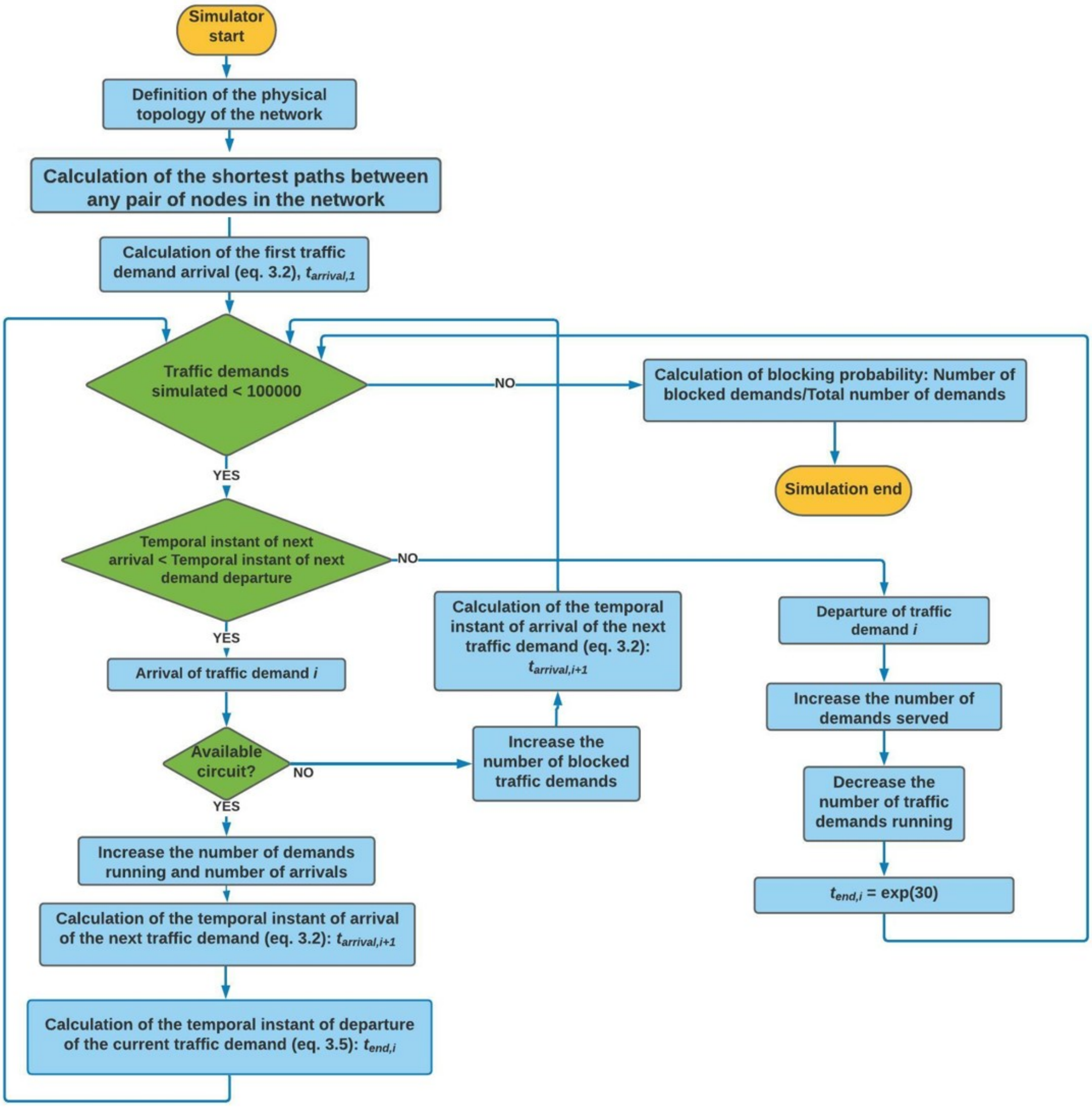
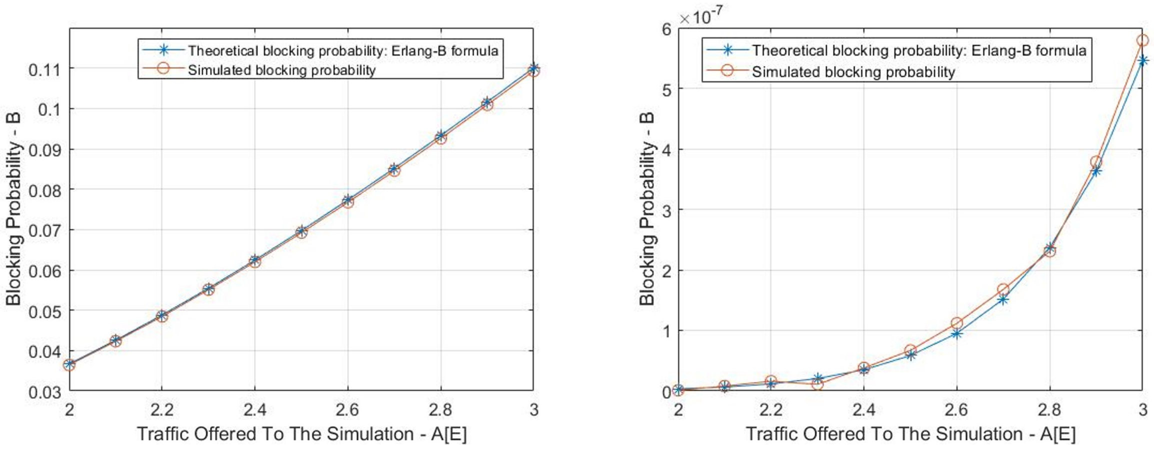

Simulator start
Definition of the physical topology of the network
Calculation of the shortest paths between any pair of nodes in the network
Calculation of the first traffic demand arrival (eq. 3.2), $t_{arrival,1}$
Traffic demands simulated < 100000
NO
Calculation of blocking probability: Number of blocked demands/Total number of demands
Simulation end
YES
Temporal instant of next arrival < Temporal instant of next demand departure
NO
Calculation of the temporal instant of arrival of the next traffic demand (eq. 3.2): $t_{arrival,i+1}$
Departure of traffic demand i
YES
Arrival of traffic demand i
Increase the number of demands served
Available circuit?
NO
Increase the number of blocked traffic demands
Decrease the number of traffic demands running
YES
$t_{end,i}$ = exp(30)
Increase the number of demands running and number of arrivals
Calculation of the temporal instant of arrival of the next traffic demand (eq. 3.2): $t_{arrival,i+1}$
Calculation of the temporal instant of departure of the current traffic demand (eq. 3.5): $t_{end,i}$

FIGURE 3.2. Flowchart of the $M/M/1/1$ system simulation.

3.2.2. Results and Discussion

In this sub-section, the blocking probability of a $M/M/c/c$ system is analysed as a function of the offered traffic, for simulator validation and corresponding blocking probability estimation. The blocking probability is obtained with an event-based simulator described in section 3.2.1 and compared with the Erlang-B formula given by equation (3.1). To achieve accurate blocking probability results, 1000000 traffic demands are simulated, for $M/M/5/5$ system simulation, and 10000000 traffic demands for $M/M/15/15$ simulation.

Two examples with a different number of circuits, $c = 5$ and $c = 15$ are evaluated, respectively, in Figures 3.3 (A) and (B) where the blocking probability is plotted as a function of the traffic offered to the simulation, A. In Fig. 3.3 (A), due to the reduced number of available circuits to serve the traffic offered, the blocking probability is very high. The reference value is 1% for optical networks, so it can be said that the values in Fig. 3.3 (A) are high. As can be observed, the simulated and the theoretical values of the blocking probability are very similar, indicating that the simulator is well implemented for this scenario.



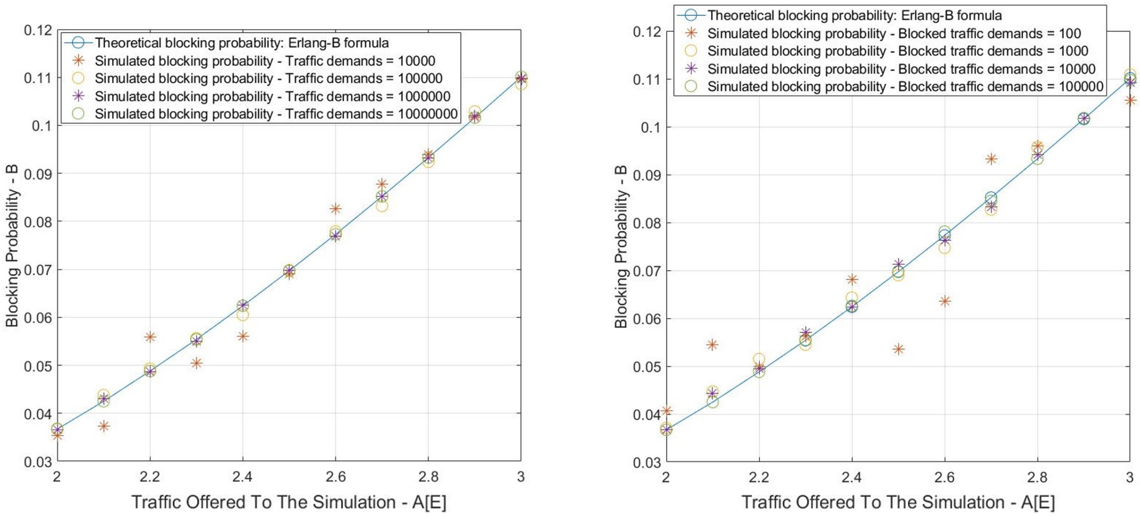
(A) $M/M/5/5$ System.

(B) $M/M/15/15$ System.

Figure 3.3. Blocking probability as a function of the offered traffic.

The dependence of the blocking probability as a function of the number of traffic demands and the number of blocked traffic demands has been analyzed, in Figs. 3.4 (A) and (B). In Fig. 3.4 (A), the theoretical values of the blocking probability obtained using the Erlang-B formula, for the $M/M/5/5$ system, are represented by blue curve. The symbols represent blocking probabilities obtained by simulation, with a variation of the number of traffic demands. In Fig. 3.4 (B), the theoretical values are represented through the blue curve. The symbols represent the blocking probabilities obtained by simulation for different numbers of blocked traffic demands.

Fig. 3.4 show that by increasing the number of simulated traffic demands, or the number of blocked traffic demands, the blocking probabilities tend to stabilize and are more approximated to the theoretical values. Hence, it can be concluded that, to reach a stabilized value of the blocking probability with sufficient accuracy, for the simulation parameters considered and for blocking probabilities between 0.01 and 0.1, it is necessary to simulate at least 1000000 traffic demands or running the simulation until 10000 blocked traffic demands are observed. Furthermore, from all the results presented in this section, it can be concluded that the simulator is validated for point-to-point $M/M/c/c$ systems.



(A) Varying the number of simulated traffic demands.

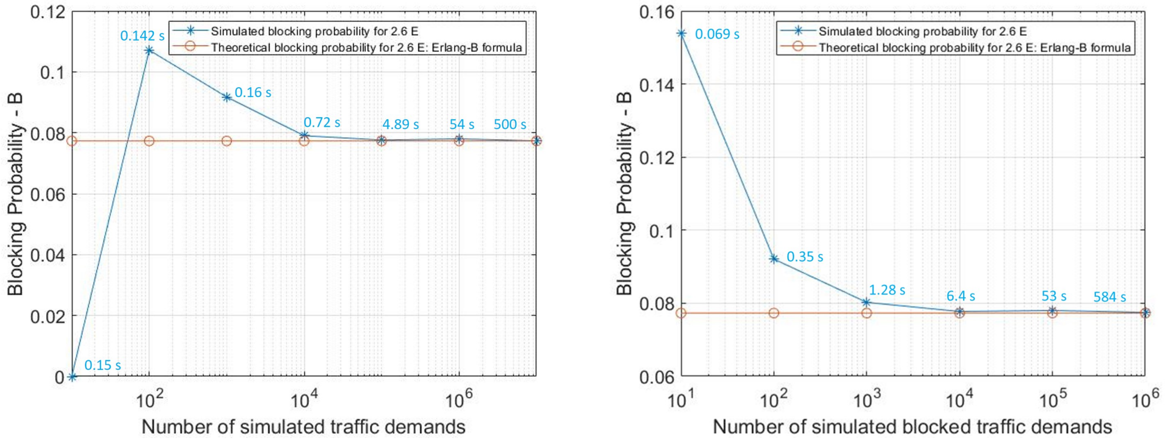
(B) Varying the number of simulated blocked traffic demands.

Figure 3.4. Blocking probabilities as a function of the offered traffic for a $M/M/5/5$ system.

In Fig. 3.5, for a theoretical blocking probability of 7.73×10^{-2} in a $M/M/5/5$ system, the simulation between 2 nodes is represented as a function of the number of simulated traffic demands and of the blocked traffic demands. To achieve the theoretical blocking probability, with a relative error of 3.75×10^{-3} using eq. (3.6), at least 100000 traffic demands must be simulated (Fig. 3.5 (A)), which corresponds a simulation time of 4.9 seconds, using a computer with an Intel(R) Core(TM) i7-8700 CPU@3.20 GHz and 16 GB of RAM. To achieve the reference blocking probability, at least 10000 blocked traffic demands must be simulated (Fig. 3.5 (B)), which corresponds to a simulation time of 6.4 seconds. The relative error of the blocking probability is obtained by

$$E_{Rx} = \frac{x - \overline{x}}{\overline{x}}, \tag{3.6}$$

where x is the theoretical blocking probability and $\overline{x}$ is the simulated blocking probability.



(A) Variation as a function of the simulated traffic demands.

(B) Variation as a function of the blocked traffic demands.

FIGURE 3.5. Variation of the blocking probability for 2.6 E of offered traffic, leading a blocking probability of 7.73×10^{-2} and simulation times.

3.3. Network with Dynamic Traffic

In the previous section, a point-to-point link with several circuits and dynamic traffic has been simulated assuming a $M/M/c/c$ model. The goal of this section is to develop a simulator that is capable of assessing the blocking probability in a dynamic traffic network composed by various nodes and links. In an optical network, the nodes are typically ROADMs and the links are implemented with optical fibers, which have typically 96 available WDM channels (C-band with 50 GHz spaced channels) [1]. The blocking probability as a function of the number of wavelengths will be assessed for various network topologies and various RWA algorithms. In this section, the criterion for stopping the simulation, and achieve a sufficiently accurate blocking probability is based on the number of blocked traffic demands, similarly to what is done for estimating the bit error rate (BER) in simulation [12]. An analytical formalism will also be introduced to validate the developed simulator, through the calculation of the blocking probability of a path.

3.3.1. Simulation Model for a Network with Dynamic Traffic

The routing algorithm considered in the simulator corresponds to the Fixed-Alternate Routing (FAR) algorithm. With this algorithm, the two shortest disjunct edge paths (the main and alternative paths) are considered for each pair of nodes. These paths are disjoint, so that, if blocking events on the two paths occur, they can be considered independent. Protection can be associated with the FAR algorithm, as this algorithm allows the calculation of two disjoint paths. In case of failure, the protection allows the selection of the first shortest disjoint path. In terms of protection, two protection architectures can be considered: 1+1 architecture and 1:1 architecture [13]. In a 1+1 architecture, a single protection path is used to protect the signal. Traffic is sent over two parallel routes, and the destination node selects the best of the two incoming signals.

In case of failure, the destination node changes to the alternate path [13]. In a 1:1 architecture, the signal is protected by a single protection path. When the primary path fails, the source and destination nodes switch to the alternate path [13].

In order to explain the developed Matlab simulator, the flowchart presented in Fig. 3.6 has been conceived.

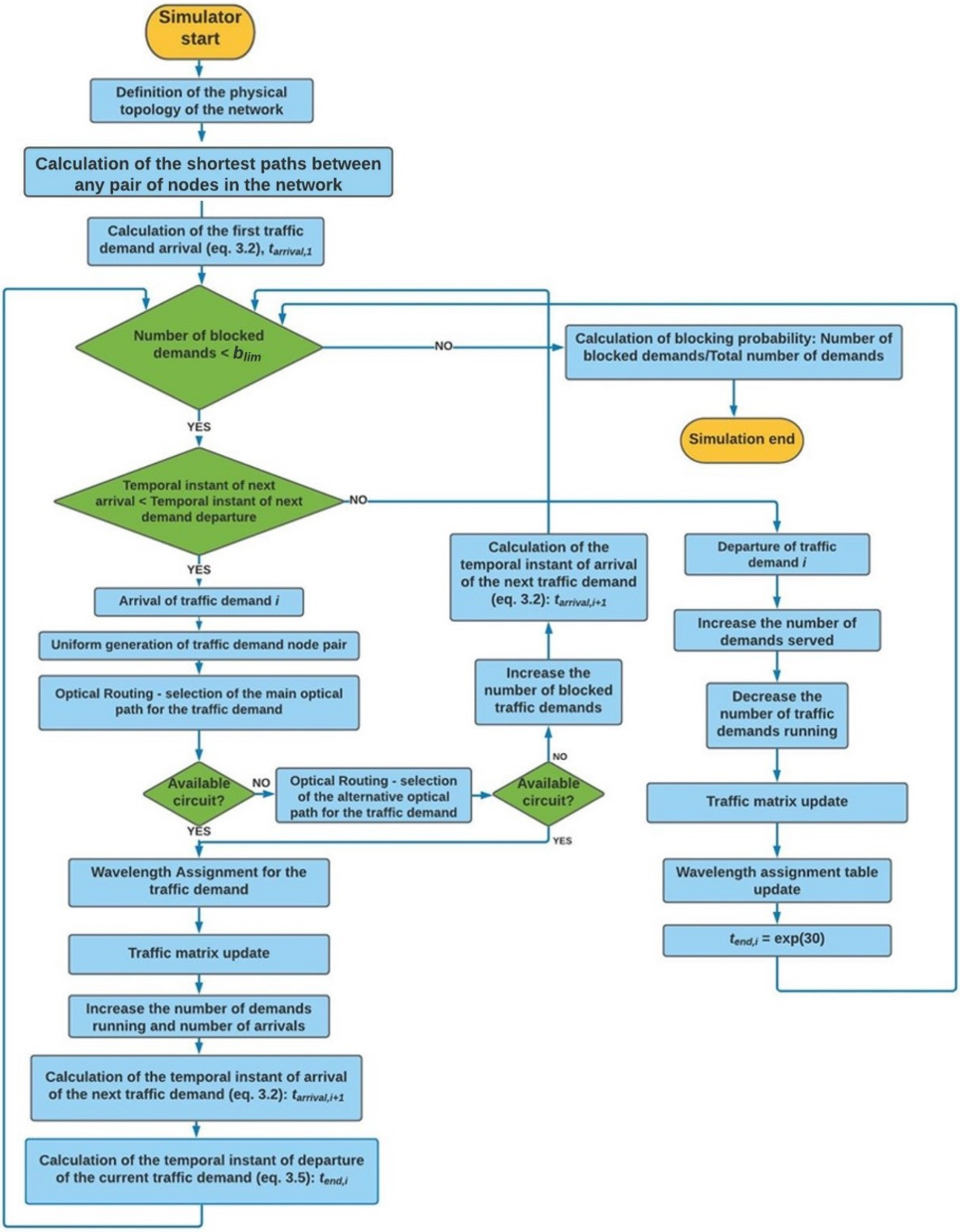

Simulator start
Definition of the physical topology of the network
Calculation of the shortest paths between any pair of nodes in the network
Calculation of the first traffic demand arrival (eq. 3.2), $t_{arrival,1}$
Number of blocked demands < b_{lim}
NO
Calculation of blocking probability: Number of blocked demands/Total number of demands
Simulation end
YES
Temporal instant of next arrival < Temporal instant of next demand departure
NO
YES
Arrival of traffic demand i
Uniform generation of traffic demand node pair
Optical Routing - selection of the main optical path for the traffic demand
Available circuit?
NO
Optical Routing - selection of the alternative optical path for the traffic demand
Available circuit?
NO
YES
YES
Calculation of the temporal instant of arrival of the next traffic demand (eq. 3.2): $t_{arrival,i+1}$
Increase the number of blocked traffic demands
Wavelength Assignment for the traffic demand
Traffic matrix update
Increase the number of demands running and number of arrivals
Calculation of the temporal instant of arrival of the next traffic demand (eq. 3.2): $t_{arrival,i+1}$
Calculation of the temporal instant of departure of the current traffic demand (eq. 3.5): $t_{end,i}$
Departure of traffic demand i
Increase the number of demands served
Decrease the number of traffic demands running
Traffic matrix update
Wavelength assignment table update
$t_{end,i}$ = exp(30)

FIGURE 3.6. Flowchart of the simulator for planning of a network with dynamic traffic and corresponding blocking probability assessment.

The model used to simulate dynamic traffic in different optical network topologies assumes that the traffic between every node pair (i.e., every source-destination pair) is uniformly distributed, the lightpath demands that arrive at each node follow a Poisson process with arrival rate λ and the lightpath service times are exponentially distributed with parameter μ [11]. The network total offered load is given by $A = \lambda/\mu$. At the beginning of the simulation, the physical topology of the network is defined, as well as the calculation of two disjoint paths between any pair of nodes. While the number of blocked traffic demands is less than the maximum number of blocked traffic demands (b_{lim}), the simulator generates a new event. If the maximum number of blocked traffic demands is reached, the simulation ends, and the blocking probability is evaluated. In this network simulation, two types of events can occur:

- the event generated is the arrival of a traffic demand. In this case, the simulator selects the optical path with the lowest cost and checks if there is a available wavelength for the selected path. If there is an available wavelength, the simulator assigns a wavelength to the path using different WA algorithms adapted for dynamic optical networks, updates the traffic matrix and calculates the arrival time of the next traffic demand (eq. 3.2) and the current traffic demand departure time (eq. 3.5). The pseudocodes of the simulated WA algorithms are represented in Appendix A. If there is no available wavelength for the main optical path, the simulator tries to assign the traffic demand to the first alternative optical path. If there is also no wavelength available for the alternative optical path, the number of blocked traffic demands is increased.

- the generated event is a traffic demand departure. In this case, the number of traffic demands served is increased, the traffic matrix and WA table are updated. In both events, the simulator returns again to the verification of the number of blocked traffic demands.

3.3.2. Analytical Formalism based on the Reduced Load Approximation

In this section, an analytical formalism to access the blocking probability in a dynamic network is used to validate the simulator in simple network scenarios, based on the reduced load approximation [14]. This approximation assumes that blocking is independent from link to link. The blocking probability of path s is given by

$$B_s = 1 - \prod_{i=1}^{K}(1 - b_l), \tag{3.7}$$

where K is the number of links of the path s and b_l is the blocking probability of the link l. The blocking probability of each link is given by the Erlang-B formula (eq. (3.1)) using

$$b_l = E_{1,N_c}(A), \tag{3.8}$$

where N_c is the number of circuits and A is the traffic intensity offered to link l, given by

$$A = \sum_{s:\, l \in R_s} a_s^l, \tag{3.9}$$

where R_s is the set of links in the path s and a_s^l is the traffic per path, given by [14]

$$a_s^l = a_s \times \frac{1}{1 - b_l} \times \prod_{K \in R_s} (1 - b_K). \tag{3.10}$$

Equation (3.10) assumes that the blocking probability of each link is not small and the traffic thinning (blocking one link depends on blocking others) in other links which is given by the term $\frac{1}{1-b_l}$ in eq. (3.10), is taken into account [14]. This approximation is used in a simple 4-link star network (node 5 is a hub node), represented in Fig. 3.7. The path blocking probability will be assessed analytically with eq. (3.7). It is considered, traffic offered to the network equal to 8 E, 7 circuits ($N_c = 7$) and $b_{lim} = 1000$.

Table 3.1 represents the simulated path blocking probability, for the 4-link star network considered in this study.

TABLE 3.1. Simulated path blocking probability, for a 4-link star network, 7 circuits, 8 E and $b_{lim} = 1000$.

Path	Blocked traffic demands	Simulated traffic demands	Blocking probability - B_s
[3,5,4]	76	13460	5.6×10^{-3}
[2,5,1]	77	13458	5.7×10^{-3}
[4,5,1]	73	13487	5.4×10^{-3}
[4,5,2]	94	13508	7×10^{-3}
[1,5,4]	82	13338	6.1×10^{-3}
[1,5,2]	89	13312	6.7×10^{-3}
[2,5,3]	78	13560	5.8×10^{-3}
[3,5,1]	98	13430	7.3×10^{-3}
[2,5,4]	82	13363	6.1×10^{-3}
[1,5,3]	91	13568	6.7×10^{-3}
[3,5,2]	74	13283	5.6×10^{-3}
[4,5,3]	86	13400	6.4×10^{-3}

All possible paths of the network under study are shown in the first column of Table 3.1. For each path, the number of simulated traffic demands is calculated for the path in question, and the corresponding number of blocked traffic demands. In this way, the computation of the simulated path blocking probability is given by

$$B_s = \frac{b_d}{N_d}, \tag{3.11}$$

where b_d is the number of blocked traffic demands and N_d is the number of simulated traffic demands. In order to validate the results of the blocking probability of the paths, presented in Table 3.1, equation (3.7) is used to compute analytically the path blocking probability. For the 4-link star network, it is important to represent the possible paths per node (3), as well as the circuits per link (7). In Figure 3.7, the 3 possible paths from node 4 to the other 3 nodes are represented.

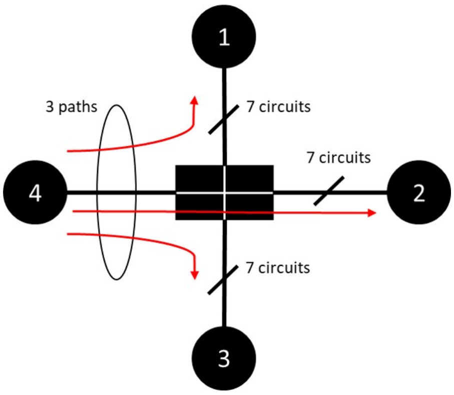


FIGURE 3.7. 4-link star network with paths per node and circuits per link.

To obtain the path blocking probability using equation (3.7), it is important to take into account some parameters, such as the number of circuits (N_c), the offered traffic to network (A), the offered traffic by node (A_{node}) and the traffic per path (A_{path}). The offered traffic per node, can be given by

$$A_{node} = \frac{A}{N_{nodes}}, \tag{3.12}$$

where N_{nodes} is the number of network nodes. With $A = 8$ E and $N_{nodes} = 4$ nodes, the offered traffic per node is $A_{node} = 2$ E. The traffic per path, can be given by

$$A_{path} = \frac{A_{node}}{p}, \tag{3.13}$$

where p is the number of paths per node. In this case, with $A_{node} = 2$ E and $p = 3$, the traffic per path is $A_{path} = \frac{2}{3}$ E. Assuming that all nodes (except the hub) send traffic to each other and the traffic per path is uniformly distributed then the blocking probability b_l is the same for all links. In this scenario and following the reduced load approximation formalism [14], the blocking probability of a link (b_1), is given by

$$b_1 = E_{1,7}(A_{path}(1 - b_1) \times 3) = E_{1,7}(\frac{2}{3}(1 - b_1) \times 3) = E_{1,7}(2(1 - b_1)). \tag{3.14}$$

Equation 3.14 must be solved iteratively, giving a link blocking probability of approximately 3×10^{-3}. Therefore, the blocking probability of a path of the 4-link star network formed by two links, is given by

$$B_s = 1 - (1 - b_1)^2 = 6 \times 10^{-3} = 0.6\%. \tag{3.15}$$

In the simulation, the blocking probabilities of the paths are between 5.4×10^{-3} and 7.3×10^{-3}, and corresponds to the average of all the probabilities presented in the fourth column of Table 3.1, because all paths are formed by two equal links, giving 0.62%, which is very close to the theoretical value given in equation (3.7) hence, validating the simulator in the case of a simple network with dynamic traffic and without WA.

In order to analyze the performance of the developed simulator, in Figure 3.8 the blocking probability as a function of the total traffic is represented, considering the simulation and the analytical expression for the 4-star network. To obtain these results, the previous equations were considered, requiring the variation of A. With this variation of traffic, the blocking probability of an optical link (b_1) is recalculated (eq. 3.14) and, consequently, the blocking probability of an optical path as well (eq. 3.15).

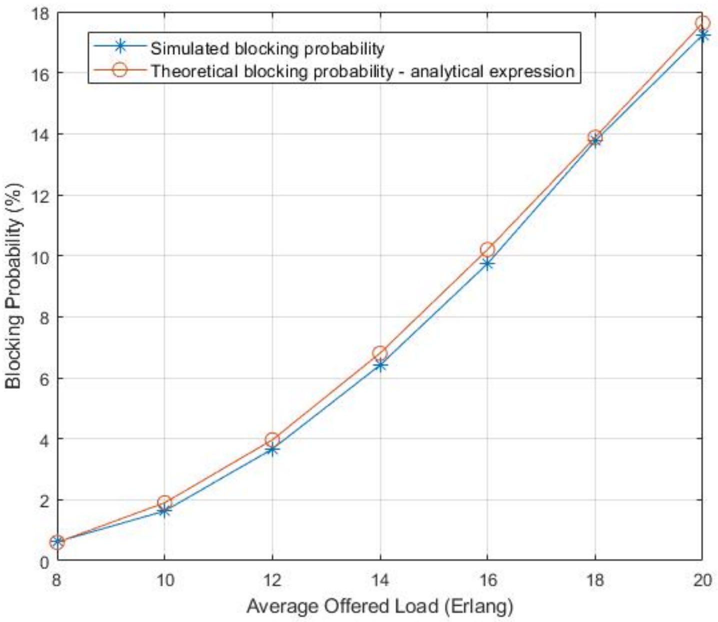


Figure 3.8. Blocking probability of an optical pah as a function of the average offered load.

3.3.3. Simulation Validation and Results

In this sub-section, the simulator presented in section 3.3.1 will be tested and validated for different network topologies and RWA algorithms, by comparison with literature results. The first network topology tested is a bidirectional ring network with 16 nodes [5], represented in Figure 3.9. The simulation parameters are described in Table 3.2.

Table 3.2. System features and parameters of the 16-node ring network [5].

Parameter	Value
Number of network nodes	16 nodes
Average offered load per node, A_{node}	{0.8, 0.9, 1, ... , 2} E
Average offered load to network, $A_{network}$	{96, 108, 120, ... , 240} E
Maximum number of blocked traffic demands, b_{lim}	1000
Routing algorithm	FAR algorithm
WA algorithm	Most-used

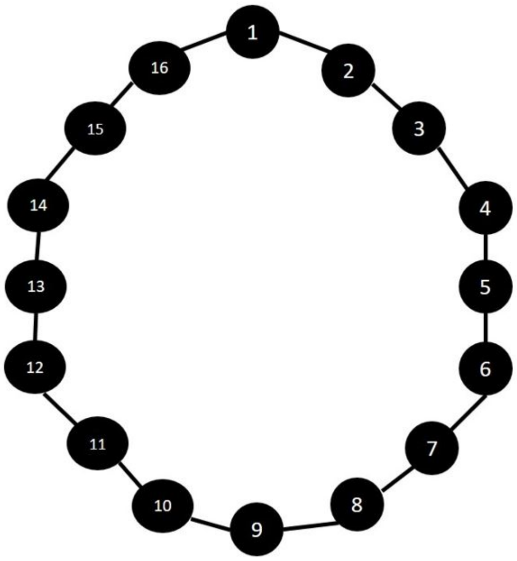


Figure 3.9. Bi-directional WDM ring network with 16 nodes.

In [5], the routing is done with the FAR algorithm and the WA algorithm considered is the Most-used, considering 48, 56 and 64 wavelengths in each optical link. The average offered load per node (A_{node}) varies between 0.8 and 2 E, which corresponds to an offered network load ($A_{network}$) that varies between 96 and 240 E, according to [15]

$$A_{network} = \frac{N_{nodes}(N_{nodes} - 1)}{2} \times A_{node}, \tag{3.16}$$

where N_{nodes} is the number of network nodes. A total mesh logical topology has been considered. Figure 3.10 represents the blocking probability of the 16-node ring network as a function of the average offered load per node considering 48, 56 and 64 wavelengths (N_c) in each fiber link. The blocking probability represented in Figure 3 of [5] is also represented for comparison purposes. For low traffic values, the blocking probability of a traffic demand is very low, as there are a large number of available wavelengths. When the offered traffic increases, the WA rate does not keep pace with the arrival rate of traffic demands and, therefore, the number of blocked traffic demands increases. The blocking probabilities obtained by the simulator developed are in very good agreement with the results presented in [5], indicating that the developed simulator and the blocking probability assessment are well implemented. Furthermore, the results shown in Fig. 3.10 indicate the correctness of the Most-used algorithm implementation.

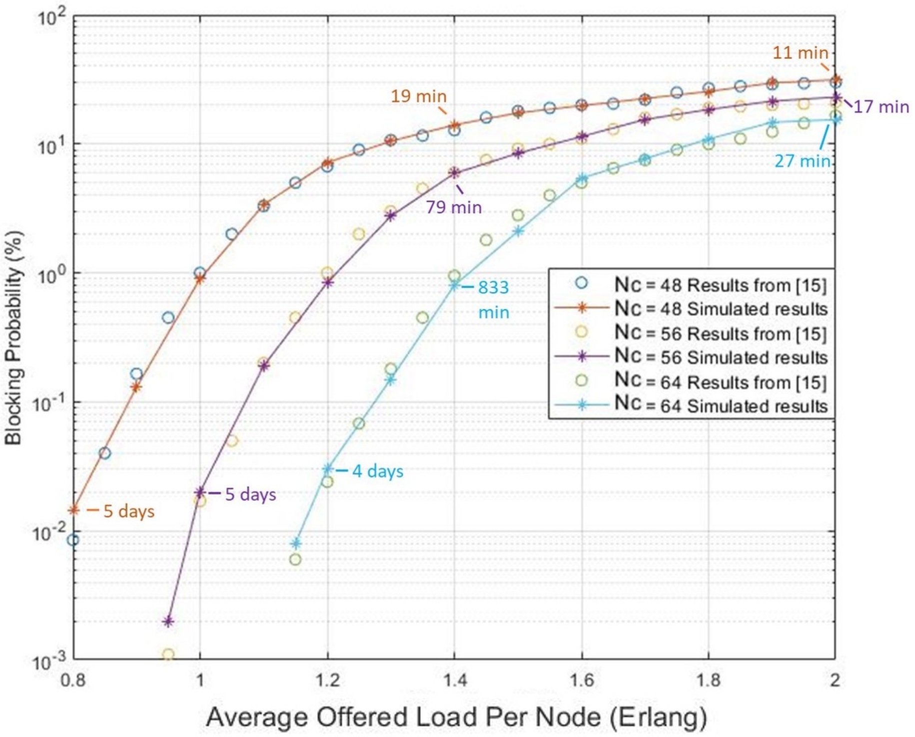


FIGURE 3.10. Blocking probability as a function of the offered load per node for a bidirecctional 16-node ring network with 48, 56 and 64 wavelengths.

In Figure 3.10, some values of the simulation time, for different blocking probabilities, are also presented. The simulation time is very dependent on the blocking probability and on the maximum wavelengths per link, to be estimated. For example, to reach a blocking probability of 10^{-2}%, the simulation takes more than 4 days, requiring 10^7 simulated traffic demands, while for a blocking probability of 1%, it takes several hours, requiring 10^5 traffic demands.

In order to verify that the network offered load is equal to the offered load imposed in the simulation, the simulated network offered load is estimated by varying the number of blocked traffic demands, using

$$A_{simulated} = \frac{N_d}{T_{sim}}, \tag{3.17}$$

where T_{sim} is the simulation time, from the arrival time of the first demand until the time of arrival of the last traffic demand. The result of this study is illustrated in Figure 3.11.

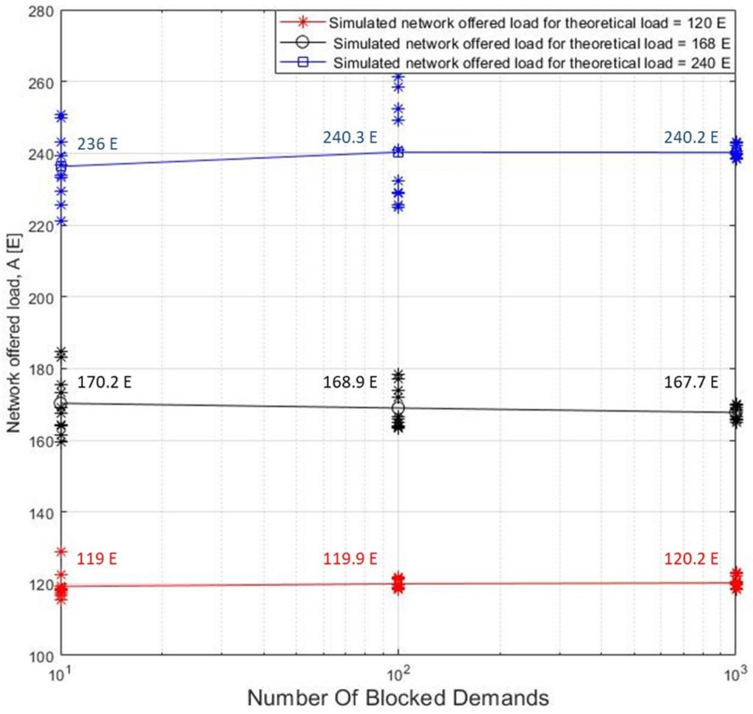
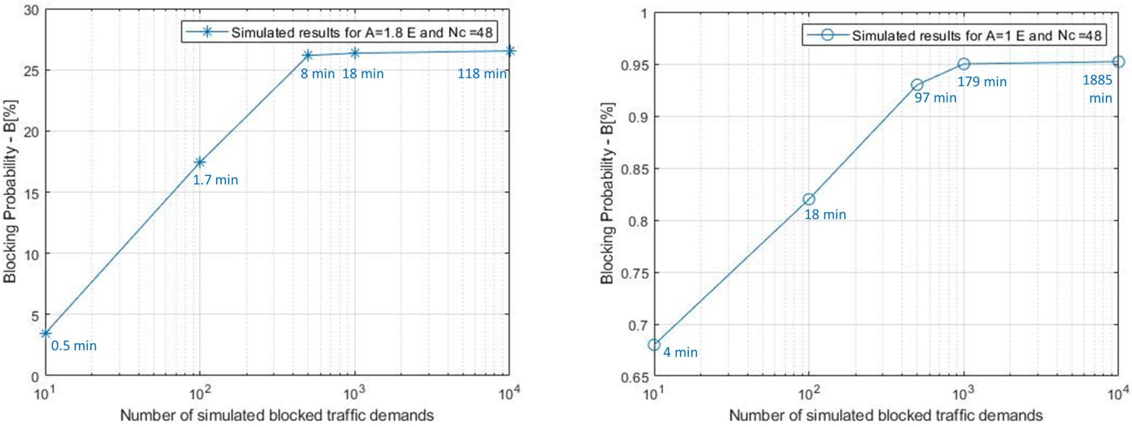


FIGURE 3.11. Simulated network offered load as a function of the number of blocked traffic demands for different theoretical network offered loads.

For each of the three theoretical network offered loads studied, ten simulation runs have been performed for $b_{lim} = 10, 100$ and 1000 blocked traffic demands. The calculation of the simulated network offered load is obtained using the average of the estimated network offered load in each run. As can be seen from Figure 3.11, the average value of the simulated network offered load tends more closely to the theoretical values after 1000 blocked traffic demands. By increasing the number of traffic demands, the variance of the simulated network offered load values tends to decrease, increasing the simulation accuracy. These results indicate the correct implementation of the dynamic network simulation tool, because the service and arrival rates imposed as simulation parameter have been confirmed, showing that at least 1000 blocked demands should be set as a stopping criterion, to guarantee a reduced error in the arrival and service rates.

For 1.8 E and 1 E of offered traffic per node and 48 wavelengths per link, the dependence of the blocking probability on the number of blocked traffic demands is shown in Figure 3.12 (A) and (B), respectively. For each value of simulated traffic demands, the average values of the blocking probabilities obtained in 10 simulation runs are represented.



(A) Average offered traffic per node equal to 1.8 E.

(B) Average offered traffic per node equal to 1 E.

Figure 3.12. Blocking probability as a function of the simulated blocked traffic demands, for a bidirecctional 16-node ring network with $N_c = 48$.

By analyzing Figures 3.12 (A) and (B), it can be concluded that the minimum number of blocked traffic demands, to achieve a stabilized blocking probability, corresponds to 500 blocked traffic demands. For this reason, for all the simulations carried out, 1000 blocked traffic demands are considered as the stopping criterion. By simulating a low number of blocked traffic demands, for lower blocking probabilities (Figure 3.12 (B)), it is possible to reach a value close to the stabilized value of the blocking probability. On the contrary, when it is necessary to achieve higher blocking probabilities (Figure 3.12 (A)), it is mandatory to simulate the largest possible number of blocked traffic demands, in order to obtain accurate blocking probabilities. The time values represented under the results, represent the average time of the 10 simulation runs.

To confirm the implementation of the Monte Carlo simulator for estimating the blocking probability in dynamic scenarios, for other situations studied in the literature, a bidirectional ring network with 8 nodes, as shown in Fig. 3.13 [6], has been considered. The system features and parameters for this case are indicated in Table 3.3.

Table 3.3. System features and parameters of the 8-node ring network [6].

Parameter	Value
Number of network nodes	8 nodes
Average offered load per node, A_{node}	{2.8, 3.2, 3.6, 4, 4.4} E
Average offered load to network, $A_{network}$	{80, 90, 100, 110, 120} E
Maximum number of blocked traffic demands, b_{lim}	1000
Routing algorithm	FAR algorithm
Wavelengths in each optical link, N_c	40
WA algorithm	First-fit, Most-used, Greedy and Random

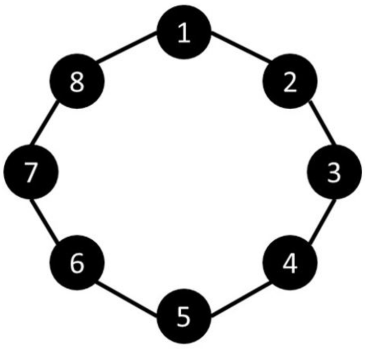


Figure 3.13. Bi-directional WDM ring network with 8 nodes.

With First-fit as WA algorithm, two scenarios of network offered traffic are studied, 90 and 100 E, which correspond respectively to a simulated blocking probability of 0.73% and 2.23%. The results from [6] indicate for these two network loads, the blocking probabilities of 0.75% and 2.5%, respectively. In addition, other offered traffic values are used to obtain blocking probabilities, shown in Fig. 3.14 proving, once again, that by increasing the offered traffic to the network, the blocking probability of a traffic demand tends to increase. Once again, the simulated results are very similar with the reference results [6], validating the dynamic network simulator and the First-fit implementation.

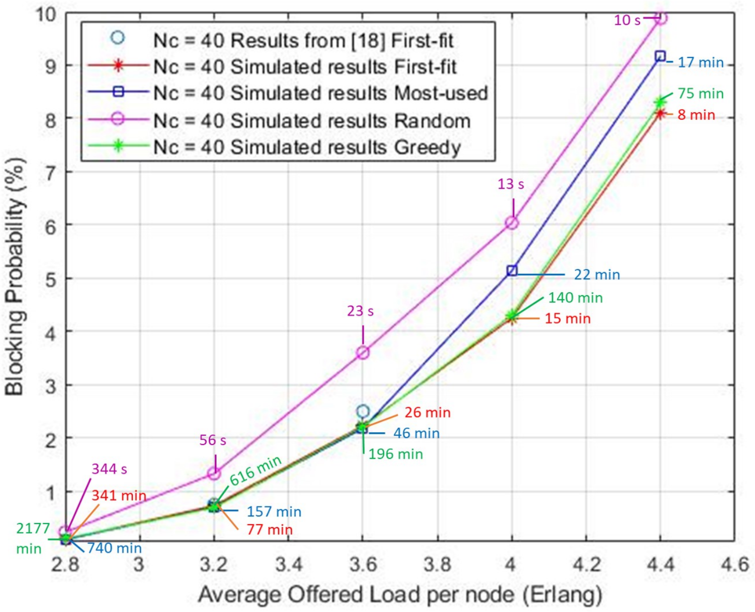


Figure 3.14. Blocking probability as a function of the offered load per node, for a 8-node ring network with 40 wavelengths.

To compare the results produced by the dynamic network simulation tool, the 8-node ring network was also studied, considering others WA algorithms, like Most-used, Greedy and Random. From Fig. 3.14, it can be seen that Random is the least advisable WA algorithm to apply in this network planning, because it produces the highest values of blocking probability of a traffic demand.

Regarding the Greedy algorithm, taking into account that the graph coloring is performed in order of arrival of the traffic demands, in practice it is similar to the First-fit algorithm, as the traffic demands are not ordered according to some strategy. The path vertex coloring is done considering the available wavelength with the smallest index, without intersecting with adjacent paths. Thus, it is expected blocking probabilities equals to the blocking probabilities obtained by the First-fit, which is visualized in Fig. 3.14. Regarding the Most-used algorithm and looking at Fig. 3.14, the First-fit and the Greedy algorithm allows to minimize the blocking probability compared to Most-used, for the highest values of average offered load per node. As happened for the 16-node ring network, also for the 8-node ring network, the WA algorithm that produces the highest blocking probability of a traffic demand corresponds to the Random algorithm.

The time value represented under the results, represents the simulation time of the simulation, using a computer with an processor Intel(R) Core(TM) i7-8565 CPU @ 1.80 GHz and 16 GB of RAM. As can be seen in Figure 3.14, the WA algorithm that calculates the blocking probability of a traffic demand more quickly corresponds to the Random algorithm, because this is the simplest algorithm, with the least computational complexity. In relation to other WA algorithms, it appears that the First-fit is the one with the lowest simulation times, about half of the simulation times obtained by the Most-used algorithm, because Most-used needs to account for each traffic demand arrival, the count of the number of links per wavelength. The algorithm that presents the longest simulation times corresponds to the Greedy graph coloring, due to high computational complexity that it presents, recurrent of the characteristics of the algorithm presented in chapter 2.

The third network tested is a real network, known as 14-node National Science Foundation Network (NSFNET) [6], represented in Fig. 3.15 and considering the system features indicated in Table 3.4, to compare the simulated blocking probabilities with the values in [6] for First-fit algorithm.

Table 3.4. System features and parameters of the NSFNET network [6].

Parameter	Value
Number of network nodes	14 nodes
Average offered load per node, A_{node}	{2.03, 2.09, 2.14, 2.19, 2.25, 2.31, 2.36} E
Average offered load to network, $A_{network}$	{185, 190, 195, 200, 205, 210, 215} E
Maximum number of blocked traffic demands, b_{lim}	1000
Routing algorithm	FAR algorithm
Wavelengths in each optical link, N_c	40
WA algorithm	First-fit, Most-used, Greedy and Random

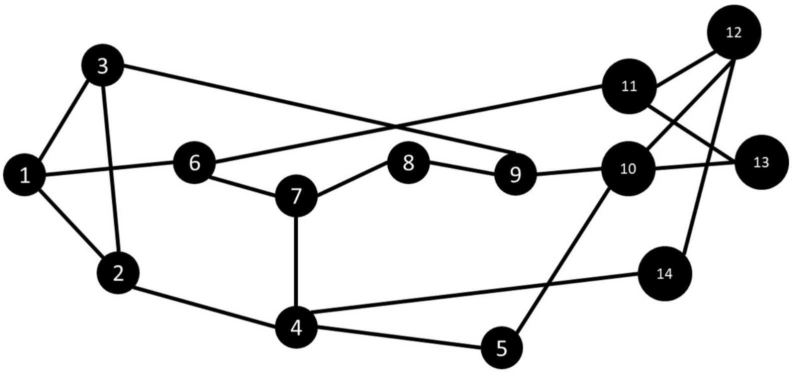


FIGURE 3.15. 14-node NSFNET network.

Fig. 3.16 represents the blocking probability of the 14-node NSFNET network as a function of the average offered load per node in Erlang. The blocking probabilities extracted from Figure 5 of [6], considering First-fit, are also represented for comparison purposes. As in the simulations of the ring networks of 8 and 16 nodes, also in the simulation of the network of 14 nodes, the simulated results of the blocking probability of a traffic demand are very similar to the reference values [6], considering First-fit algorithm, hence validating the simulator for a more complex real network. The Random algorithm was also simulated for this network and as expected, the blocking probability of a traffic demand are superior compared to First-fit, Most-used and Greedy.

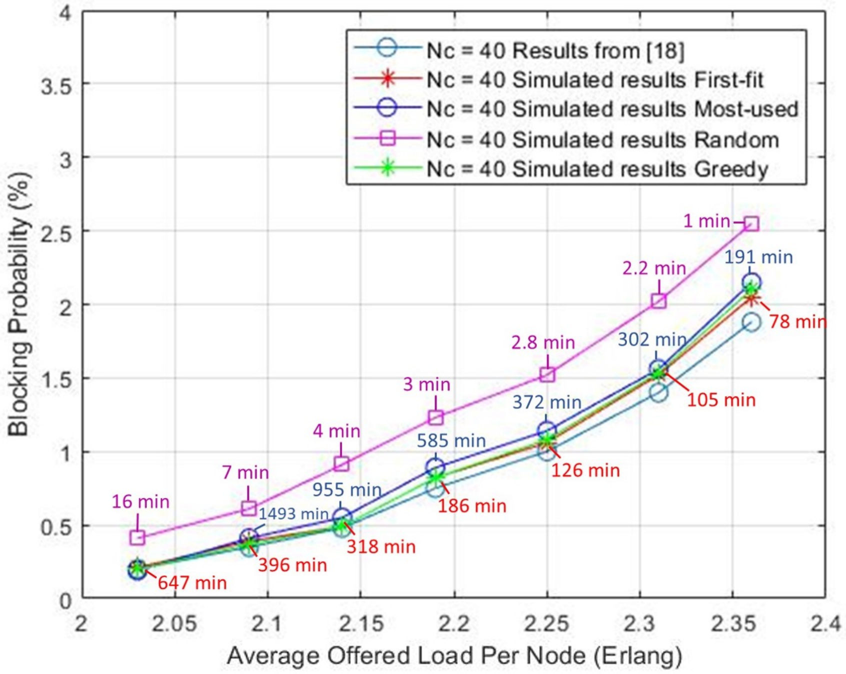


FIGURE 3.16. Blocking probability as a function of the average load offered per node, for a NSFNET 14 network with 40 wavelengths.

In each simulated blocking probability, in Fig. 3.16, the simulation time is also presented, which is dependent of the number of simulated traffic demands, that is, when more traffic demands are simulated, biggest is the simulation time for planning each demand. As analyzed in ring networks, also for the NSFNET network, Random is the WA algorithm with the shortest simulation times, because it is a simple algorithm, without any kind of computational complexity. Of the remaining WA algorithms, the one that produces the shortest simulation times is the First-fit, with the Greedy algorithm being the most time-consuming.

Another of the real networks tested, corresponds to the British Telecom (BT) of the United Kingdom core network with 22 nodes [16]. This network topology is represented in Figure 3.17 and for the simulation studies performed, the system parameters indicated in Table 3.5 are used.

TABLE 3.5. System features and parameters of the BT 22 nodes network [16].

Parameter	Value
Number of network nodes	22 nodes
Average offered load per node, A_{node}	{0.6, 0.7, ..., 3} E
Average offered load to network, $A_{network}$	{138.6, 161.7, ..., 693} E
Maximum number of blocked traffic demands, b_{lim}	1000
Routing algorithm	FAR algorithm
Wavelengths in each optical link, N_c	40
WA algorithm	First-fit, Most-used, Greedy and Random

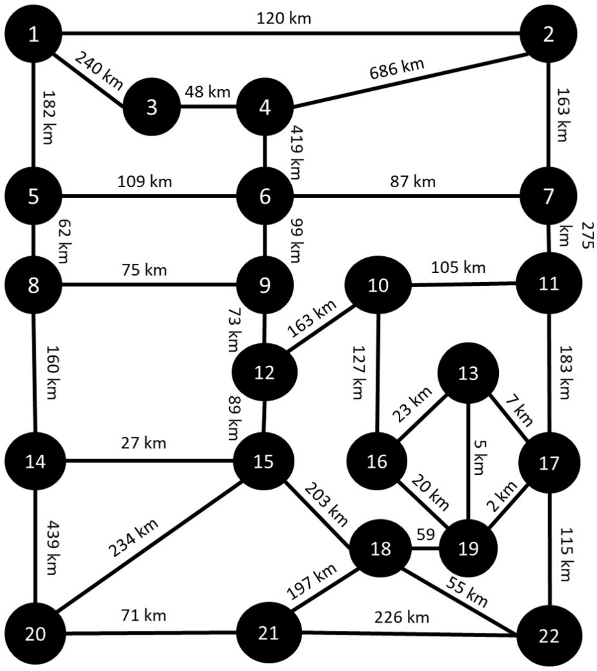


FIGURE 3.17. The BT 22 node network.

Fig. 3.18 represents the blocking probability of the British Telecom network obtained by simulation as a function of the average offered load per node in Erlang.

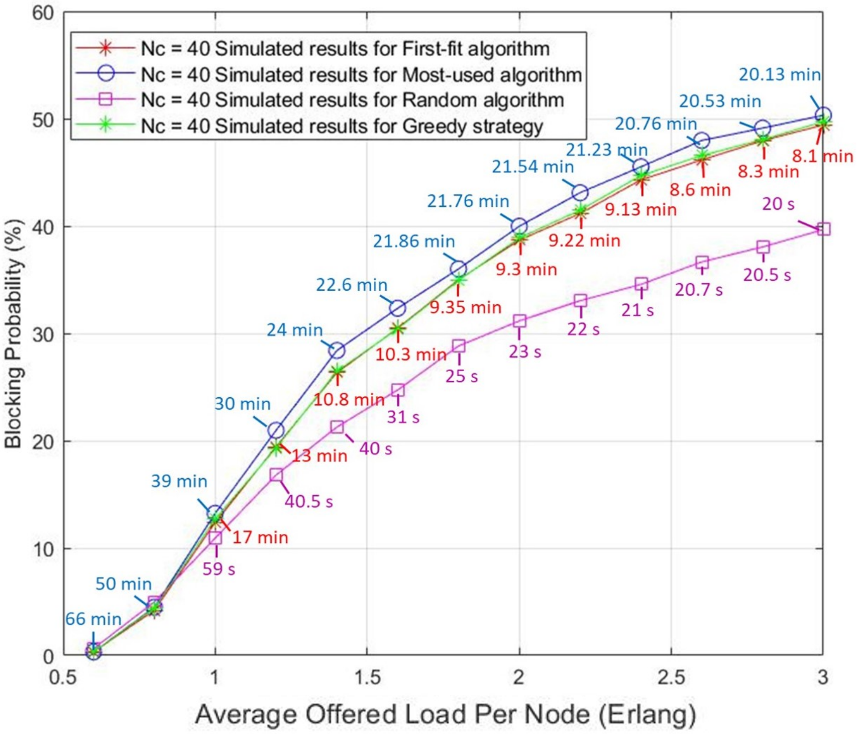


Figure 3.18. Blocking probability as a function of the average load offered per node, for the BT 22 nodes network with 40 wavelengths. The corresponding simulation computational times are also presented.

With the simulation of the British Telecom network with 22 nodes it is shown once again that the Most-used algorithm tends to use a greater number of wavelengths, compared to the First-fit algorithm, which leads to a slightly higher blocking probability. It is also visible that the simulation time of the First-fit algorithm is shorter, when compared to the Most-used algorithm, as it does not need to keep an account of the number of links used per wavelength. Once again, it is verified that the Greedy graph coloring strategy leads to blocking probabilities very similar to the ones obtained with the First-fit algorithm. In addition, the Random algorithm is also simulated for the same network. In this case, one wavelength is selected randomly and assigned to the lightpath. For this network, unlike the previously simulated optical networks, the calculated blocking probability of a traffic demand, using the Random algorithm as the WA algorithm, is lower than the blocking probability of a traffic demand considering First-fit, Most-used or Greedy strategy of graph coloring, for blocking probabilities above 10%. Therefore, for this BT with 22 nodes network with dynamic traffic scenario, the WA algorithm that leads to the lowest blocking of traffic demands with a much reduced computational time is the Random algorithm. For blocking probabilities below 10%, the performance of the various WA algorithms is very similar. In order to analyze with more detail the performance of the WA algorithms for lower values of traffic offered per node in the BT network, Figure 3.19 has been obtained.

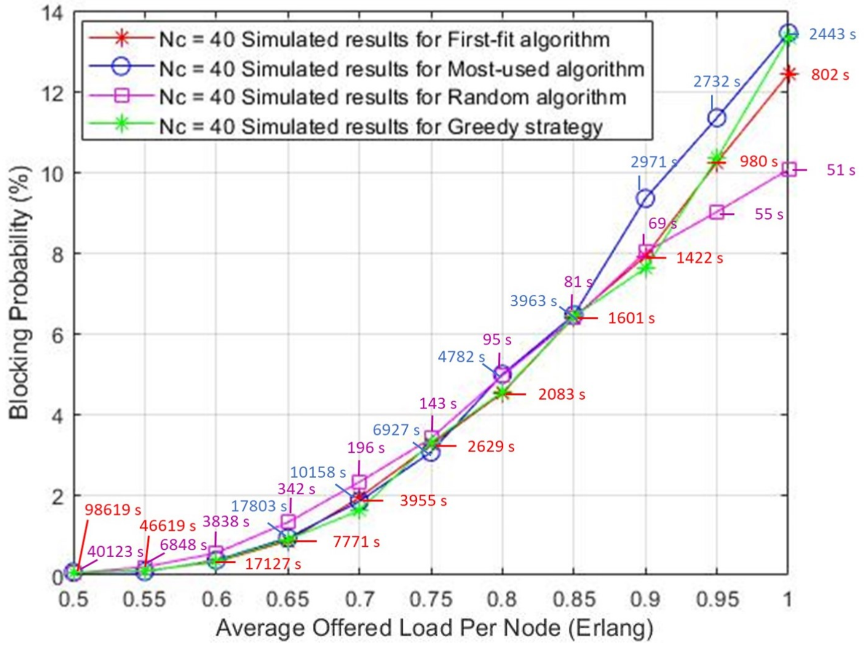


Figure 3.19. Blocking probability as a function of the lower average load offered per node, for the BT 22 nodes network with 40 wavelengths.

From Figure 3.19, it can be seen that up to 0.9 E of traffic offered per node, the Random algorithm produces higher blocking probabilities, compared to the other WA algorithms. For higher values of traffic offered by nodes, above 0.9 E, as also seen in Figure 3.18, the Random algorithm starts to bring advantages, as it leads to lower blocking probabilities in a significantly shorter computational time.

3.4. Conclusions

With the $M/M/c/c$ system implemented for dynamic traffic, it was possible to validate the model through two system examples: $c = 5$ and 15. This validation was important to ensure the correct implementation of the dynamic optical network. The validation was done by comparing it with the theoretical results obtained through the Erlang-B formula.

After the validation of the network dynamism implementation, the RWA algorithms in dynamic optical networks were implemented and validated, as well as the calculation of the blocking probability of a traffic demand, in different network topologies. As a routing algorithm, the FAR algorithm was implemented, which allows obtaining the two shortest disjoint paths between any pair of nodes in the network. As WA algorithm, First-fit, Most-used, Random and Greedy algorithms were implemented. These algorithms were used to assign wavelengths to traffic demands, in optical networks such as 16-node ring network, 8-node ring network, NSFNET network and BT network. For the different networks, the different WA algorithms were simulated, and the results of the blocking probability of a traffic demand are represented in figures.

To validate the simulation of the WA algorithms, reference blocking probabilities [5] were considered for the 16-node ring network and using the Most-used algorithm as the

WA algorithm. A simulation of the system was performed under the same conditions as [5], with the results obtained (blocking probabilities of a traffic demand) by the simulator (Figure 3.10) being quite identical to the theoretical results. An identical validation was also performed taking into account [6], in which the First-fit algorithm was considered. This simulation of the system was made using the conditions of [6], where it was possible to obtain very identical results to those presented in [6]. Thus, the simulator was validated. In general, the WA algorithms (from among the algorithms simulated in this section) that allow lower blocking probabilities of a traffic demand correspond to the First-fit and Greedy algorithms. Conversely, the Random algorithm calculates the highest blocking probabilities of a traffic demand, with the Most-used algorithm in an intermediate situation. For the BT network with 22 nodes, due to the high number of distinct optical paths for higher traffic offered to the network, the WA algorithm that calculates lower blocking probabilities is the Random algorithm.

In algorithms like First-fit and Greedy graph coloring, the wavelengths are indexed and for the optical path of the traffic demand, the selection of the wavelength with the lowest index is attempted before trying to select a wavelength with a higher index. By assigning wavelengths in this manner, existing connections are compressed into a smaller number of total wavelengths. Thus, it is possible to leave a greater number of wavelengths available for other traffic demands and, consequently, minimize the blocking probability of a traffic demand. In the Most-used algorithm, the most used wavelength in the rest of the network is selected, trying to maximize the reuse of the wavelength in the network. For this procedure, the algorithm requires global knowledge of the network. This algorithm, unlike the First-fit and Greedy algorithms, does not minimize the blocking probability of a traffic demand as it does not guarantee the maximum number of available wavelengths for the next traffic demands.

Regarding the simulation times, obtained using an computer with an processor Intel(R) Core(TM) i7-8565 CPU @ 1.80 GHz and 16 GB of RAM, the WA algorithms that have shorter simulation times are Random, due to the simplicity of the algorithm (randomly assigned a wavelength), and First-fit, due to being assigned the wavelength available with the lowest index. The Most-used algorithm produces simulation times approximately the double of the First-fit simulation times, because it is necessary to account for the number of optical links, the number of distinct assigned wavelength, which is very time-consuming. The algorithm that produces the longest simulation times corresponds to the Greedy graph coloring strategy, because it is a conventional mathematical problem that consists of coloring all the nodes of a graph so that there are no two adjacent nodes with the same color/wavelength whenever an update occurs on the network. The high time consumption is due to the basic idea of the algorithm, which consists in passing the graph of the physical topology of the network $G(V, E)$ to a graph of $G(W, P)$, whose nodes are the optical paths.

CHAPTER 4

Implementation of a Dynamic Graph Coloring Algorithm for Wavelength Assignment

4.1. Introduction

In this chapter, the dynamic graph coloring algorithm presented in section 2.7.3, the Small-Bucket algorithm, is studied for WA in the context of planning dynamic optical network. In section 4.2, the Small-Bucket is explained and a small example of how it works is discussed. In section 4.3, the simulation model for the Small-Bucket is presented through a flowchart. In section 4.4, the graphical user interface developed for the simulator is explained. In sections 4.5, 4.6 and 4.7, the Small-Bucket algorithm is studied for WA in three different network topologies, 16 and 8-node ring and US Backbone Network (UBN). The blocking probability, number of recolorings, number of colors used and simulation times are assessed and compared for the referred topologies. Moreover, these results are compared with those of the First-fit. Finally, in section 4.8, the conclusions are presented.

4.2. Small-Bucket Algorithm

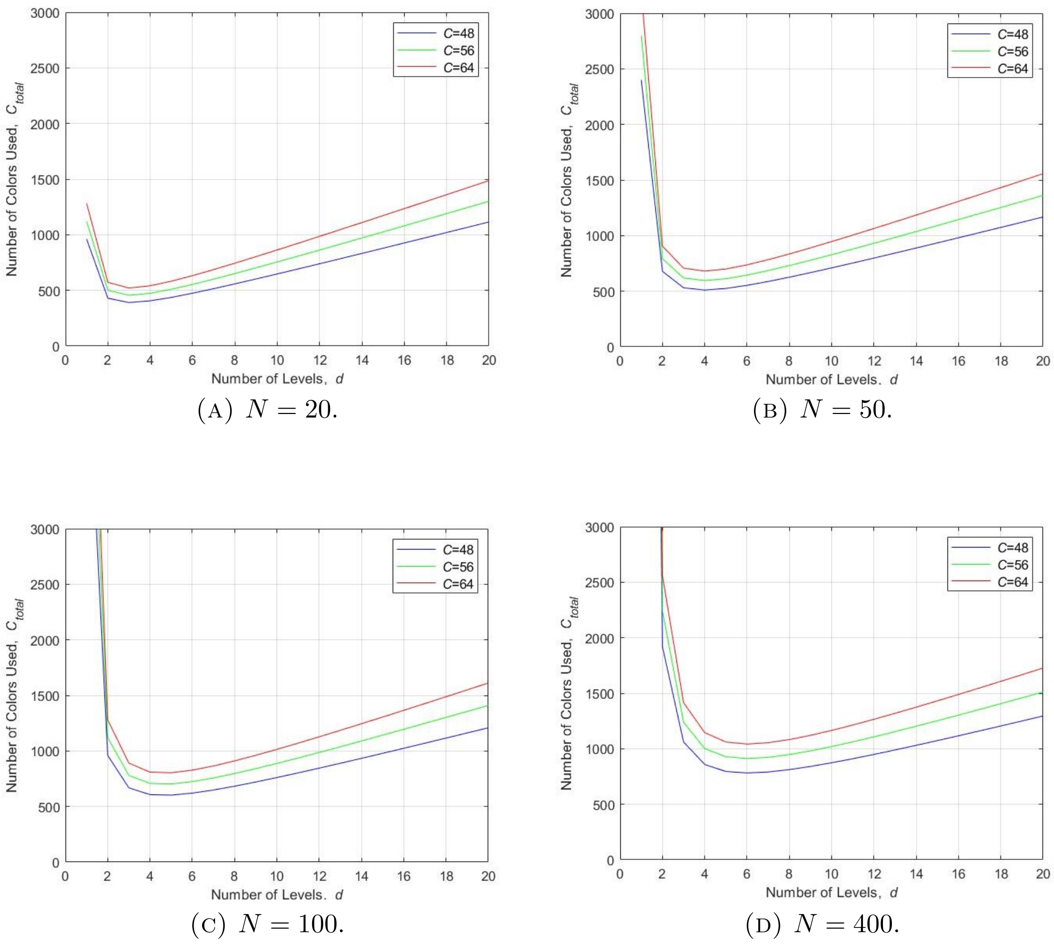
The Small-Bucket algorithm is a graph coloring algorithm [3] that is going to be applied and tested in this chapter for WA in dynamic networks. In this algorithm, a number of buckets is used and inside each bucket, several traffic demands can be accommodated. Each bucket has its own set of colors, so that two adjacent vertices do not have the same color. As a rule, there is a sequence of buckets of increasing size, organized by d levels, each containing s buckets, and a RESET bucket. The RESET bucket is a special bucket, where all demands are placed whenever all the levels are unable to accommodate a new demand. This bucket has an infinite capacity for vertices (i.e. demands), but like the other buckets, it has a limited number of colors (C).

The idea of the algorithm corresponds to the placement of a new demand in a bucket at level $i = 0$. If this is the last empty bucket at that level, placing it violates the rule of the algorithm: per level there must be at least one empty bucket. Considering that the level $i = 0$ does not have available buckets, all vertices are shifted to the first empty bucket of the next level and then the WA using the specific color set of this new bucket is attempted. These shifts are repeated whenever the main rule is not fulfilled, up to a limit situation where all vertices are shifted to and recolored in the RESET bucket. When these level changes occur, vertex recoloring must also occur. In optical networks, a recoloring means that a wavelength reconfiguration (assign a demand to another wavelength with laser reconfiguration) needs to be performed. The number of recolorings is an important performance metric since it is highly related to the blocking probability. More recolorings mean a decrease in the blocking probability, but it also means that more colors are used. Two limit scenarios can be considered. The first one corresponds to perform the maximum number of recolorings i.e., N recolorings per update. This is possible if, for example, the Greedy algorithm is used for each demand arrival. The second scenario corresponds to perform the minimum number of recolorings, i.e., no recolorings. In this situation a new color is assigned for each new demand and, so, the total number of colors will be increased.

The Small-Bucket algorithm just described lies between these two limiting cases. With this algorithm there are at most d recolorings per update in average [3]. As shown in [3], the total number of available colors is limited by the number of levels and the maximum number of colors used by the buckets. The total number of colors used is at most [3],

$$C_{total} = C \times d \times N^{1/d}, \tag{4.1}$$

with C being the maximum number of colors per bucket and N the maximum number of distinct vertices to color. Figure 4.1 shows the number of colors as a function of the number of levels for $C = 48, 56$ and 64 considering $N = 20$, 50, 100 and 400.



(A) $N = 20$.

(B) $N = 50$.

(C) $N = 100$.

(D) $N = 400$.

FIGURE 4.1. Theoretical maximum number of colors used by the Small-Bucket algorithm as a function of the number of levels with the number of colors per bucket as a parameter, considering (A) $N = 20$, (B) $N = 50$, (C) $N = 100$ and (D) $N = 400$.

From Figure 4.1, it can be concluded that the minimum of the total number of colors slightly increases from 3 to 6 levels, respectively, for $N = 20$ and $N = 400$. For high values of d, it can be observed that for all the values of N studied the theoretical maximum number of colors is between 1500 and 1700 for $C = 64$, between 1300 and 1500 for $C = 56$ and between 1100 and 1300 for $C = 48$. For low values of d, there is a huge difference in the maximum number of colors when increasing N. In the limit, for $d = 1$, the number of colors used is maximum and is given by $C \times N$. The variables used by the algorithm are presented in Table 4.1.

TABLE 4.1. Variables considered in the Small-Bucket algorithm.

Variable	Designation	Value
N	Maximum number of vertices to color in the path graph representing the optical network logical topology	–
N_R	Number of vertices during last RESET	–
s	Number of buckets per level	$\lceil N_R^{1/d} \rceil$
d	Number of levels	–
b_{lim}	Maximum number of blocked demands	1000
i	Index of selected level to check availability	$\{0, 1, ..., d-1\}$
n	Index of bucket selected to check availability	$\{0, 1, ..., s-1\}$
d_i	Level selected to check availability	$\{d_0, d_1, ..., d_{d-1}\}$
$s_{n,i}$	Bucket n of level i selected to check availability	$\{s_{0,i}, s_{1,i}, ..., s_{s-1,i}\}$
C	Maximum number of colors per bucket	–

The Small-Bucket algorithm will be described next through an example, considering a ring with 5 nodes. In this network, it is possible to have a maximum of $N = 20$ distinct paths, considering a full mesh logical topology. The number of levels considered is $d = 1$. The algorithm uses ds buckets, grouped into d levels of s buckets each. All buckets at level i $(0 \leq i < d)$ have capacity for s^i vertices. The RESET bucket has a maximum number of colors equal to C. While running the algorithm, each level should have always at least one empty bucket. Blue colors are used for level $i = 0$ and green colors are assumed for the RESET bucket. Considering these parameters, the initialization layout of the Small-Bucket is shown in Figure 4.2. The value on top of the buckets represents the capacity of each bucket.

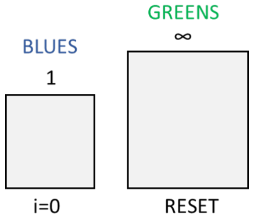
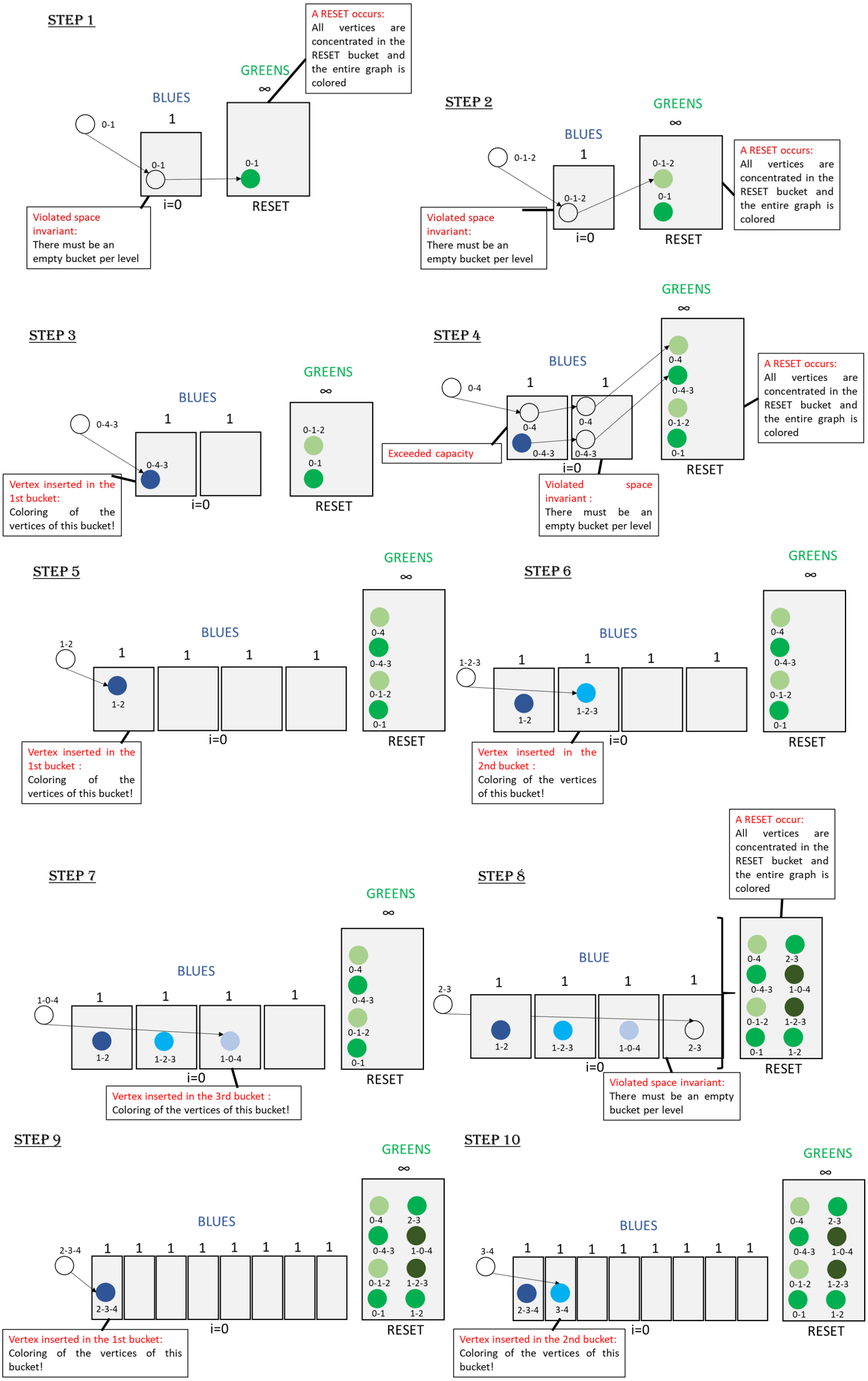


FIGURE 4.2. Small-Bucket algorithm - buckets initialization layout.

Figure 4.3 represents the example of the Small-Bucket algorithm for WA using the initial configuration represented in Figure 4.2 and considering the 10 first traffic demands.



A RESET occurs: All vertices are concentrated in the RESET bucket and the entire graph is colored

Violated space invariant: There must be an empty bucket per level

Violated space invariant: There must be an empty bucket per level

A RESET occurs: All vertices are concentrated in the RESET bucket and the entire graph is colored

Vertex inserted in the 1st bucket: Coloring of the vertices of this bucket!

Exceeded capacity

Violated space invariant : There must be an empty bucket per level

A RESET occurs: All vertices are concentrated in the RESET bucket and the entire graph is colored

Vertex inserted in the 1st bucket : Coloring of the vertices of this bucket!

Vertex inserted in the 2nd bucket : Coloring of the vertices of this bucket!

A RESET occur: All vertices are concentrated in the RESET bucket and the entire graph is colored

Vertex inserted in the 3rd bucket : Coloring of the vertices of this bucket!

Violated space invariant: There must be an empty bucket per level

Vertex inserted in the 1st bucket: Coloring of the vertices of this bucket!

Vertex inserted in the 2nd bucket: Coloring of the vertices of this bucket!

FIGURE 4.3. Small-Bucket algorithm example: insertion of vertices for $N = 20$ possible optical paths, considering the 10 first traffic demands.

In the traffic demands represented in Fig. 4.3, there is a violation of the main rule in steps 1, 2, 4 and 8. In the remaining, it is possible to place the new vertex in the first empty bucket at level $i = 0$ and perform the respective coloring. In general, if the last bucket of the last level is filled, the system is redefined, by emptying each bucket and placing the demands in the RESET bucket, with a new coloring (steps 1, 2, 4 and 8). Each bucket has its own set of colors, so that two adjacent vertices do not have the same color. This idea can be verified in demand #2, in which the vertices 0-1-2 and 0-1 are colored in the RESET bucket, with different colors. In some steps of the example, as from step 2 to 3, there is a change in the number of buckets per level. As in the previous step, the system has been redefined and the RESET bucket is completely filled, when demand #3 arrives, the number of buckets in level $i = 0$ is recalculated, taking into account the number of vertices in the last RESET, N_R. Thus, there is the addition of one bucket in level $i = 0$, in the transition of steps 2 to 3. The number of buckets at level $i = 0$ is also increased from steps 4 to 5 and 8 to 9, in the example represented in Fig. 4.3.

Besides the example just described, we have also a video demonstration of how the Small-Bucket algorithm works. The video can be found in [4] and considers a ring network with 16 nodes. The first 42 traffic demands are analyzed in this video demonstration.

4.3. Simulation Model of the Small-Bucket Algorithm

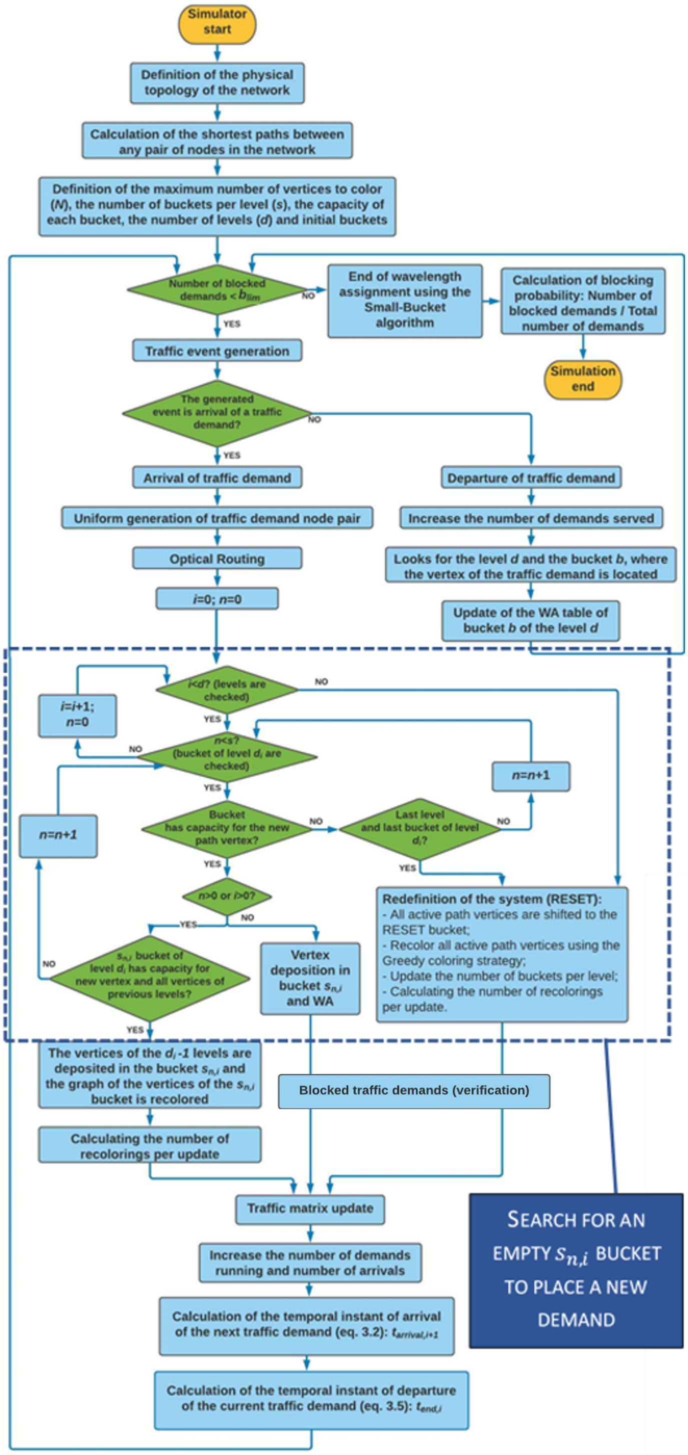
The developed Matlab simulator for the Small-Bucket algorithm described in section 4.2, can be explained through the flowchart presented in Fig. 4.4. Before generating the traffic events, the algorithm is initialized:

(1) definition of the maximum number of distinct vertices (paths) to color (N);
(2) definition of the number of buckets per level (s);
(3) definition of the capacity of the buckets in each level (s^i);
(4) setting the number of levels (d).

The algorithm is executed while the number of blocked demands does not reach b_{lim}. If the limit is reached, the blocking probability of a traffic demand is calculated and the simulation ends. While the limit is not reached, traffic events are generated. In this simulation, two types of events can occur:

- arrival of a traffic demand. The buckets of the d levels are checked, in ascending order of n, to find the first bucket with available capacity. If the level of the selected bucket with capacity for the vertex is level 1, the new vertex is placed in a bucket of level 1 and WA occurs, assigning the available wavelength with the lowest index (an example of this coloring is the 3rd traffic demand arrival exemplified in Fig. 4.3). If the level of the selected bucket with available capacity has an index greater than 1, the vertex and all the vertices of the previous levels are moved to the selected bucket, the wavelengths are reassigned according to the Greedy algorithm on the selected bucket. Finally, if there is no level capable of receiving the vertex, a system RESET occurs (an example is the eighth traffic demand arrival exemplified in Fig. 4.3). With a system RESET, all demands are moved and colored/recolored in the RESET bucket, the number of buckets per level (s) is updated and the number of recolorings per update is calculated. Whenever there are recolorings it is important to check if traffic demands are blocked. In these recolorings, if there is no wavelength available for an already served traffic demand, that traffic demand is blocked, which is a situation that should not happen in a real situation.

- departure of a traffic demand. The bucket where the vertex of the corresponding path is located is searched. Once found, that path is removed from the bucket.



Simulator start
Definition of the physical topology of the network
Calculation of the shortest paths between any pair of nodes in the network
Definition of the maximum number of vertices to color (N), the number of buckets per level (s), the capacity of each bucket, the number of levels (d) and initial buckets
Number of blocked demands < b_{lim}
NO
End of wavelength assignment using the Small-Bucket algorithm
Calculation of blocking probability: Number of blocked demands / Total number of demands
Simulation end
YES
Traffic event generation
The generated event is arrival of a traffic demand?
NO
YES
Arrival of traffic demand
Departure of traffic demand
Uniform generation of traffic demand node pair
Increase the number of demands served
Optical Routing
Looks for the level d and the bucket b, where the vertex of the traffic demand is located
i=0; n=0
Update of the WA table of bucket b of the level d
i<d? (levels are checked)
NO
i=i+1; n=0
YES
NO
n<s? (bucket of level d_i are checked)
n=n+1
YES
n=n+1
Bucket has capacity for the new path vertex?
NO
Last level and last bucket of level d_i?
NO
YES
YES
n>0 or i>0?
NO
YES
Redefinition of the system (RESET):
- All active path vertices are shifted to the RESET bucket;
- Recolor all active path vertices using the Greedy coloring strategy;
- Update the number of buckets per level;
- Calculating the number of recolorings per update.
NO
$s_{n,i}$ bucket of level d_i has capacity for new vertex and all vertices of previous levels?
Vertex deposition in bucket $s_{n,i}$ and WA
YES
The vertices of the d_i-1 levels are deposited in the bucket $s_{n,i}$ and the graph of the vertices of the $s_{n,i}$ bucket is recolored
Blocked traffic demands (verification)
Calculating the number of recolorings per update
Traffic matrix update
SEARCH FOR AN EMPTY $s_{n,i}$ BUCKET TO PLACE A NEW DEMAND
Increase the number of demands running and number of arrivals
Calculation of the temporal instant of arrival of the next traffic demand (eq. 3.2): $t_{arrival,i+1}$
Calculation of the temporal instant of departure of the current traffic demand (eq. 3.5): $t_{end,i}$

FIGURE 4.4. Flowchart of the simulator for WA using Small-Bucket.

4.4. Graphical User Interface for the Implemented Simulator

To facilitate the usage of the simulator developed in section 4.3, a Graphical User Interface (GUI) was developed. The GUI developed to implement the dynamic optical network planning tool allows specifying several parameters such as the routing algorithm, the WA algorithm, the physical network topology, traffic offered per node, service rate and maximum number of wavelengths per link. Figure 4.5 represents a screenshot of the GUI of the simulator, showing the WA algorithms and the possible physical network topologies that can be chosen.

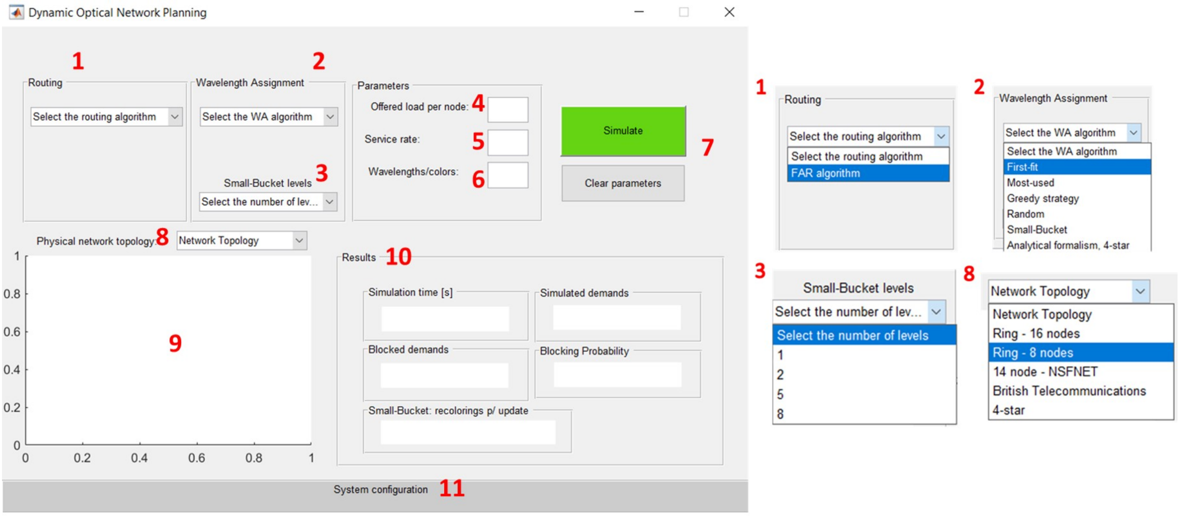


FIGURE 4.5. GUI of the simulator.

TABLE 4.2. Description of the menus considered in the GUI.

Menu	Designation	Description
1	Routing	Selection of the simulated routing algorithm
2	Wavelength Assignment	Selection of the simulated WA algorithm
3	Small-Bucket levels	If the Small-Bucket algorithm is selected, this menu should be used to select the number of levels
4	Offered load per node	Selection of the average traffic offered to the network by each node
5	Service rate	Selection of network demand service rate
6	Wavelengths/colors	Selection of the maximum number of wavelengths/colors for each optical link
7	Simulate and clear parameter	Button that runs the simulation and button that allows the reset of the simulator, respectively
8	Physical network topology	Selection of the network physical topology considered for the simulation
9	Physical topology graph	Graphical presentation of the network considered for the simulation
10	Simulation results	Printing simulation results
11	Simulation state	Describes the state of the simulation

Table 4.2 shows the description of each program menu. With this GUI of the dynamic optical network planning simulator, it is much easier to run all the simulations described in this work, as well as changing other parameters not shown. To validate the analytical formalism presented in section 3.3.2, it is necessary to select ”Analytical formalism, 4-star” in the Wavelength Assignment menu and ”4-star” in the Physical topology menu.

4.5. Small-Bucket Algorithm Results and Discussion for the 16-node Ring Network

In this section, the simulator presented in section 4.3 will be tested and compared with First-fit algorithm for the 16-node ring network. In particular, the blocking probability, the total number of colors, recolorings and the computation time will be assessed.

The simulation parameters are described in Table 3.2, but now the WA algorithm is the Small-Bucket. The number of system levels is assumed to be $d = \lfloor \ln(N) \rfloor = \lfloor \ln(240) \rfloor = 5$, because this number of levels gives the minimum number of colors used, as discussed in the section 4.2 (see Figure 4.1). In the simulations carried out, the maximum number of colors per bucket considered is C = 48, 56 and 64 colors. These values were considered so that the maximum number of colors per bucket is equal to the maximum number of wavelengths per link in the First-fit simulation considered in section 3.3.3.

4.5.1. Blocking Probability and Simulation Time

In this section, the blocking probability and the simulation time will be analyzed. Fig. 4.6 represents the blocking probability as a function of the average offered load per node considering C = 48, 56 and 64 colors and d = 5. The blocking probabilities calculated using the First-fit algorithm are also represented for comparison purposes. Furthermore, some values of the simulation times are also represented, in Fig. 4.6.

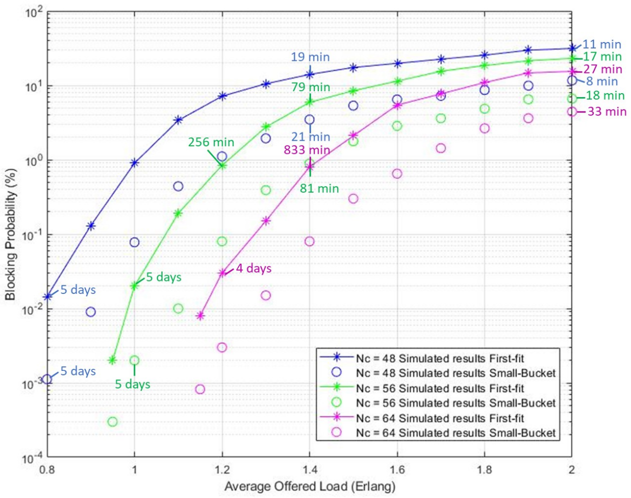


FIGURE 4.6. Blocking probability as a function of the average offered load per node for a bidirecctional 16-node ring network with C = 48, 56 and 64 colors per bucket considering the Small-Bucket as WA algorithm. The First-fit is also represented for comparison purposes.

As can be seen from Figure 4.6, the Small-Bucket with C colors per bucket gives a lower blocking probability, compared to the First-fit algorithm with $N_c = C$. The difference between the obtained blocking probabilities, for an average load of 2 E, is approximately 8% in favour of the Small-Bucket, due to the fact that the Small-Bucket uses a much higher number of colors to color the vertices partitioned by s buckets. With this number of levels, the vertices can be placed in different graph coloring subspaces, thus allowing to minimize the blocking probability of a traffic demand. The comparisons between the First-fit algorithm assuming N_c wavelengths per optical link and the Small-Bucket algorithm considering C colors per bucket and a total of C_{total} colors is not completely fair, because the Small-Bucket requires a very high number of colors, according to equation (4.1).

Figure 4.6 represents also some examples of the simulation time for different average offered traffic per node and considering the Small-Bucket and First-fit algorithms. Comparing the First-fit algorithm with the Small-Bucket for the same network traffic, the simulation time obtained is very similar. For example, when $A_{node} = 1.4$ E, the First-fit algorithm needs a simulation time of 19 minutes, while the Small-Bucket algorithm needs 21 minutes. However, the Small-Bucket is much faster than the First-fit to achieve a target blocking probability. For example, to achieve a blocking probability of 1% for $C = 56$, the Small-Bucket takes 81 minutes ($A_{node} = 1.4$ E), while the First-fit takes 256 minutes ($A_{node} = 1.2$ E). The simulation time of the Small-Bucket algorithm is influenced by two major events: vertex shifts between buckets (about 20% of simulation time) and WA in a selected bucket (about 80% of simulation time).

Assuming First-fit with $N_c = 48$ as reference, in Fig. 4.7, the blocking probability as a function of the offered load per node with the maximum number of colors per bucket ($C = 20, 40, 45$ and 48) as a parameter considering the Small-Bucket is shown. The blocking probability obtained with the First-Fit is also represented for comparison purposes.

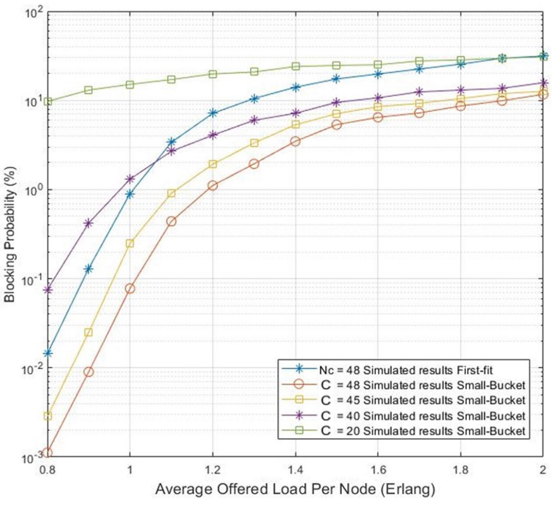


FIGURE 4.7. Blocking probability as a function of the average offered load per node, with the maximum number of colors per bucket in the Small-Bucket algorithm as a parameter, considering the 16-node ring network.

The goal is to check the maximum number of colors per bucket, that produces similar results to the ones obtained using First-fit with $N_c = 48$. From Fig. 4.7, it can be observed that the maximum number of colors per bucket that produces such results, considering the First-fit with $N_c = 48$ wavelengths per link, corresponds to $C = 20$ for high traffic (> 1.4 E) and $C = 40$ for lower values of traffic (< 1.4 E).

Figure 4.8 represents the blocking probability as a function of the average offered load per node with the number of levels d as a parameter considering $C = 48$. The blocking probability obtained with the First-Fit is also represented for comparison purposes.

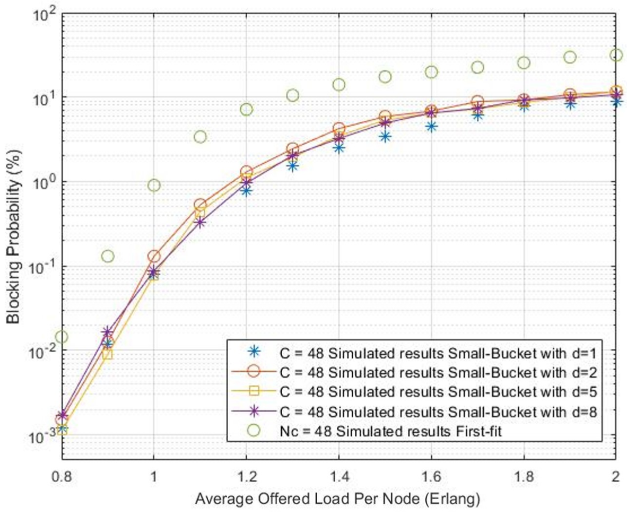


FIGURE 4.8. Blocking probability as a function of the average offered load per node, with the number of levels in the Small-Bucket algorithm as a parameter, considering the 16-node ring network.

From Fig.4.8, one can conclude that the number of levels practically does not influence the blocking probability. So, it can be concluded that having more levels (e.g. $d = 8$) with few buckets per level is similar to having less levels (e.g. $d = 1$) with more buckets per level, in what concerns the blocking probability.

4.5.2. Total Number of Colors Used and Number of Recolorings per Update

In this subsection, the total number of used colors and the total number of recolorings per update is studied for the 16-node ring network.

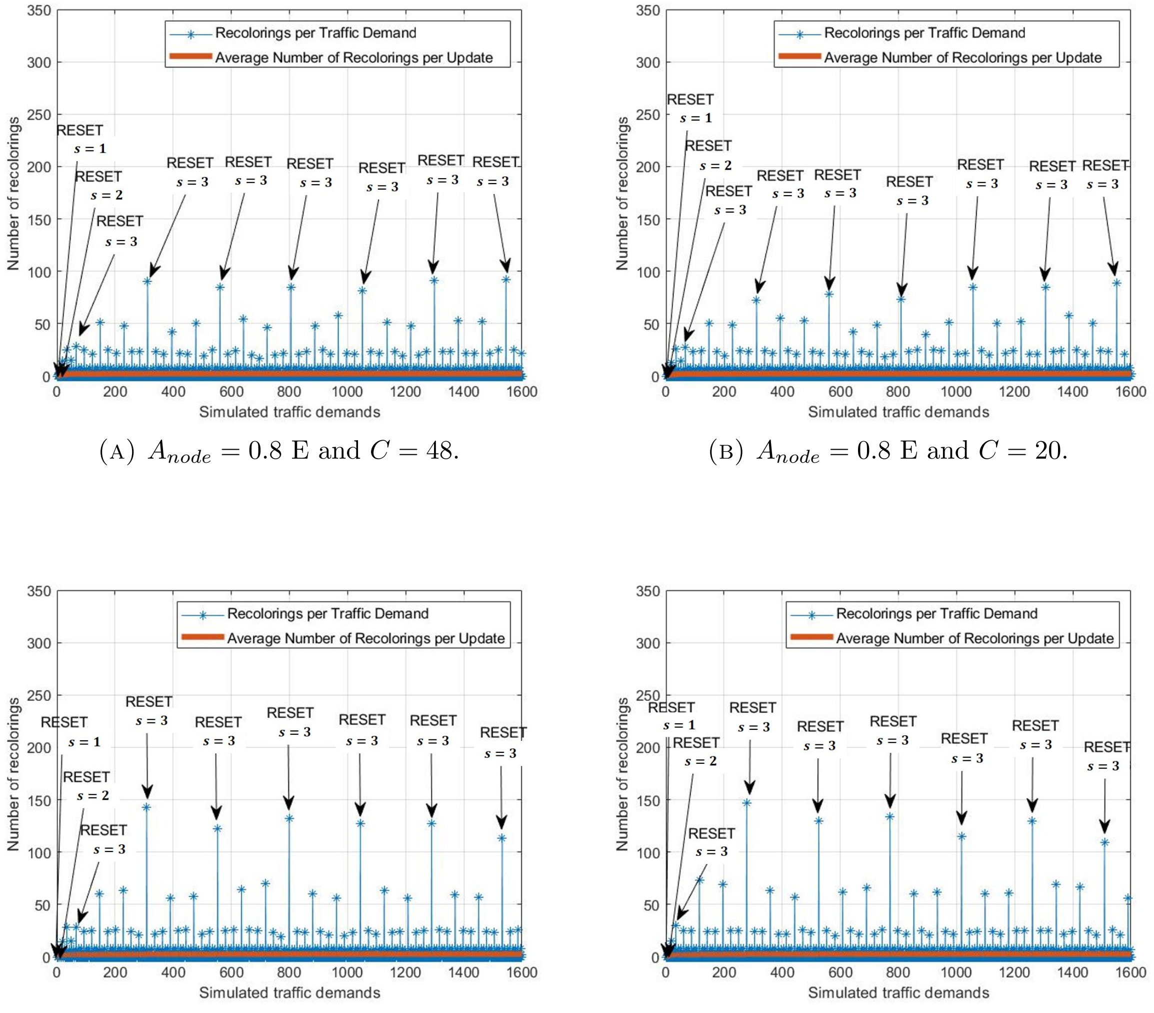
The number of recolorings per update consists of the number of wavelengths reassigned in active traffic demands after the arrival of a new demand. Recolorings occur in the RESET bucket and when there are level changes (vertex shifts between/within levels). Table 4.3 represents the average number of recolorings per update as a function of the average offered load per node, with the number of levels, d, as a parameter ($d = 1, 2, 5$ and 8). These average number of recolorings per update are obtained after 1000 blocked traffic demands are reached.

Table 4.3. Recolorings per update for the 16-node ring network.

		$C = 48$	$C = 56$	$C = 64$
Average recolorings per update with $d = 1$	$A_{node} = 1.4$ E	0.26	0.16	0.15
	$A_{node} = 1.8$ E	0.37	0.29	0.21
	$A_{node} = 2$ E	0.39	0.30	0.24
		$C = 48$	$C = 56$	$C = 64$
Average recolorings per update with $d = 2$	$A_{node} = 1.4$ E	1.34	1.32	1.32
	$A_{node} = 1.8$ E	1.36	1.34	1.33
	$A_{node} = 2$ E	1.37	1.36	1.34
		$C = 48$	$C = 56$	$C = 64$
Average recolorings per update with $d = 5$	$A_{node} = 1.4$ E	2.94	2.73	2.64
	$A_{node} = 1.8$ E	2.52	2.57	2.60
	$A_{node} = 2$ E	2.77	2.33	2.56
		$C = 48$	$C = 56$	$C = 64$
Average recolorings per update with $d = 8$	$A_{node} = 1.4$ E	3.01	3.08	3.10
	$A_{node} = 1.8$ E	3.11	3.11	3.11
	$A_{node} = 2$ E	3.14	3.14	3.14

From [3], it can be concluded that the Small-Bucket, with $d > 0$, maintains an adequate coloration of a graph by recoloring a maximum of d vertices per update, each time a new demand arrives. As can be seen from Table 4.3, this condition is fulfilled for all number of levels tested. The number of recolorings per update increases with the number of levels in the system. At the lowest tested value ($d = 1$), the average number of recolorings per update is around 0.3. In this situation, recolorings occur in vertex shifts inside buckets of the same level or in RESET situations. When eigth levels are simulated, RESETs are less frequent but recolorings occur frequently inside buckets of a particular level, giving an average of three recolorings per update.

In Figure 4.9 (A) through (D), the behavior of the number of recolorings as a function of the number of simulated traffic demands is studied, for $d = 5$ and two values of the average offered load per node, $A_{node} = 0.8$ E and $A_{node} = 2$ E, which correspond to $A_{network} = 96$ E and $A_{network} = 240$ E, respectively. Figure 4.9 (A) and (C) represent simulations with $C = 48$, while Figure 4.9 (B) and (D) correspond to $C = 20$. The average number of recolorings per update is also represented in Fig. 4.9.



(A) $A_{node} = 0.8$ E and $C = 48$.

(B) $A_{node} = 0.8$ E and $C = 20$.

(C) $A_{node} = 2$ E and $C = 48$.

(D) $A_{node} = 2$ E and $C = 20$.

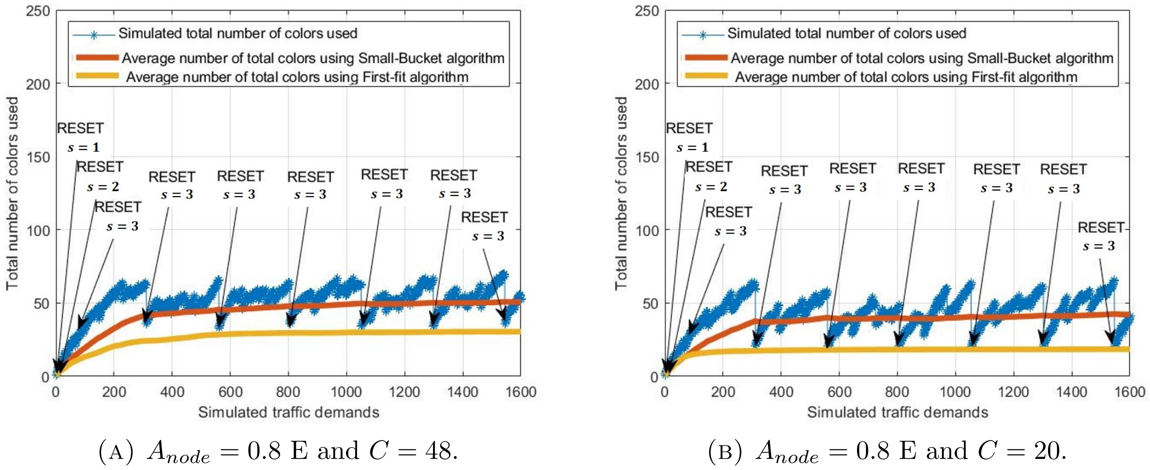
FIGURE 4.9. Simulated number of recolorings as a function of the number of traffic demands and average offered load per node for the 16-node ring network with $d = 5$.

For illustrative purposes, in Fig. 4.9 and in the following, only the first 1600 arrivals of traffic demands are shown. In this figure, the RESET situations are highlighted, as well as the evolution of the number of buckets s per level along the simulation, being this number recalculated every time a RESET occurs. For example, in Fig. 4.9 (A), the traffic demand #300 corresponds to a RESET situation. In this demand, there are approximately 90 recolorings and the number of buckets per level, after this RESET, is $s = 3$. From Fig. 4.9 (A), it can be also observed that there are 6 big RESETS, each one corresponding to approximately 90 recolorings. A big RESET is a RESET outside the initial phase of the system. The system has an initial phase in which the number of buckets per level (s) increases, in order to accommodate the average traffic on the network until a stable situation is reached, in which the number of system recolorings evolves in a relatively regular way, and the blocking probability remains approximately constant. Moreover, from Fig. 4.9 (A), it can be also observed that the number of recolorings has several floors, the highest one corresponds to the big RESETs, and the other ones correspond to the recolorings due to vertices level shifts. For example, as already said, the number of recolorings in a RESET bucket is approximately 90, whereas the number of recolorings due to level changes are approximately 50, 25, 10 and 3 (not visible in Fig. 4.9 (A)), that correspond, respectively, to changes from level 3 to 4, level 2 to 3, level 1 to 2 and level 0 to 1. In the stable phase of the system, the number of buckets per level tends to $s = 3$ and, consequently, the capacity of a bucket at level i is 3^i vertices. Hence, after the initial phase, the bucket capacity in each level is 1, 3, 9, 27 and 81, respectively, for level 0, 1, 2, 3 and 4. In Fig. 4.9 (A), the average number of recolorings is also represented, and is approximately 3, after the system initial phase. This low average number of recolorings is reached due to the high number of incoming traffic demands that are placed in the available buckets without causing any recoloring. In the scenario analyzed in Fig. 4.9 (A), 1064 traffic demands arrivals without any recoloring are counted.

In Fig. 4.9 (C), with higher average traffic ($A = 2$ E), but with the same number of colors per bucket ($C = 48$), the number of recolorings in the floors is larger than the ones found in Fig. 4.9 (A). In the RESET situations outside the initial phase, there are approximately 150 recolorings, whereas in Fig. 4.9 (A), this number is approximately 90. In situations of higher traffic, as represented in Fig. 4.9 (C), the vertex coloration rate is much higher than the departure rate of traffic demands, leading to a high bucket occupancy along the simulated traffic demands evolution.

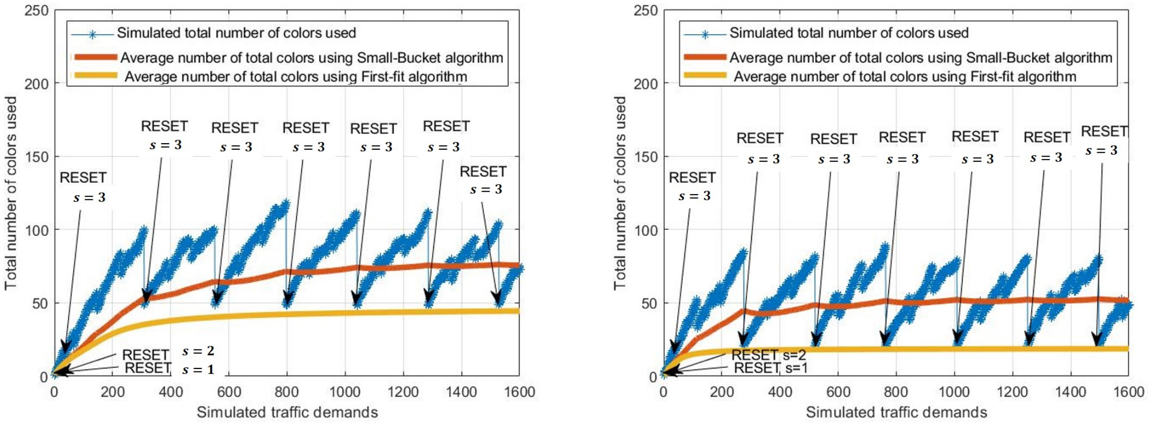
In Fig. 4.9 (B) with $C = 20$, and the same offered load than in Fig. 4.9 (A) with $C = 48$, it can be observed that the recoloring floor levels are practically of the same magnitude in both figures, and the recolorings in the RESET bucket are also 90. This is due to the fact that the buckets capacity, i.e. number of vertices, remains the same, despite the decrease in the number of colors per bucket.

Fig. 4.10 shows the total number of colors used, as well as its average value, as a function of the simulated traffic demands, for the same parameters considered in Fig. 4.9.



(A) $A_{node} = 0.8$ E and $C = 48$.

(B) $A_{node} = 0.8$ E and $C = 20$.



(C) $A_{node} = 2$ E and $C = 48$.

(D) $A_{node} = 2$ E and $C = 20$.

FIGURE 4.10. Simulated number of colors used as a function of the simulated traffic demands and average offered load per node for the 16-node ring network with $d = 5$.

In Figure 4.10, the number of colors using the First-fit with a maximum of $N_c = 48$ colors per bucket is also represented for comparison purposes. The RESET situations, as well as, the number of buckets per level s, are also highlighted. As can be observed in Figure 4.10 (A), the number of colors start to increase from 0 to 70 and then at traffic demand #300, a RESET occurs and the total number of colors decreases to 48. This decrease is due to the shift of all demands to the RESET bucket, which has at maximum only C available colors to assign (in the example, $C = 48$). After traffic demand #300, the total number of colors used seems to have a sawtooth behaviour due to the occurrence of big RESETS, which means that the number of colors varies between 48 and 70, every 220 traffic demands. In this situation, the average number of total colors used by the Small-Bucket algorithm tends to a constant value, approximately 50 colors, while the First-fit algorithm uses only an average of 25 colors. This difference is due to the fact that the First-fit algorithm only considers a single coloring space, while the Small-Bucket considers different coloring spaces, represented by the buckets.

In Fig. 4.10 (C), the average offered load per node, $A_{node} = 2$ E, is greater than in Fig. 4.10 (A), where $A_{node} = 0.8$ E. The main difference between Fig. 4.10 (A) and (C), after the algorithm initial phase, is the higher number of colors used in Fig. 4.10 (C) along the traffic demands evolution, that leads to a higher average number of colors used. When the traffic offered increases, the number of colors used is higher because the buckets are filled more quickly and, as a result, more buckets and distinct colors are used.

The number of colors as a function of the traffic demands is represented in Figures 4.10 (B) and (D), for $C = 20$. It is observed that the average number of colors is lower (approximately 50), and the sawtooth behavior has a more accentuated slope than in Figures 4.10 (A) and (C), because the number of colors per bucket is smaller ($C = 20$) and, in RESET situations, the lower number of used colors corresponds this value.

From the scenarios analyzed in Figures 4.9 and 4.10 it can be concluded that the Small-Bucket algorithm produces a large number of recolorings, that would require fast tunable transceivers, in order to tune both the transmitter and receiver for the new wavelength, which brings, besides technological issues related to the tuning speed, cost and management issues, that were not relevant when the WA is performed with, for example, the First-fit algorithm [7].

In the study performed in this section, a maximum total traffic of 240 E is used, and assuming a typical service duration in dynamic optical networks of 500 ms [7], then the arrival rate of demands is 192 demands/s, which gives a 2.1 ms time interval between demands. Knowing, also, that there is on average 1 recoloring per 3 demands, then the reconfiguration time of the lasers should be several orders of magnitude lower than 6.3 ms (the time between 3 demands) [7]. So, assuming 3 orders of magnitude lower, a reconfiguration time of 6.3 μs should be used, which can be difficult to achieved with current tunable transceiver technology [18]. In [18], reconfiguration times of 30 μs are reported.

In all results presented in this subsection, it has been observed that the number of colors and recolorings is not affected by the blocked traffic demands. This happens because blocked demands are never placed inside the buckets.

In Fig. 4.7, the Small-Bucket algorithm gives a blocking probability similar to the one given by the First-fit algorithm ($N_c = 48$) when $C = 40$ and $A_{node} = 1.1$ E. For this scenario, in Fig. 4.11, the total number of colors used as a function of the simulated traffic demands is represented.

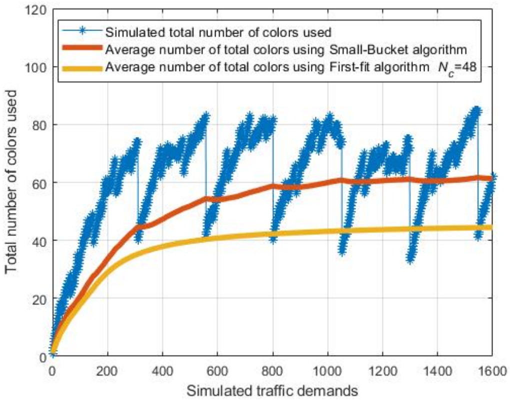


FIGURE 4.11. Simulated number of colors used as a function of the simulated traffic demands and average offered load per node for the 16-node ring network with $d = 5$ and $A_{node} = 1.1$ E.

With this study, it is observed that to achieve the same blocking probability, the Small-Bucket algorithm uses on average a greater number of colors (approximately 12) compared to the First-fit algorithm.

4.6. Small-Bucket Algorithm Results and Discussion for the 8-node Ring Network

In this section, the 8-node ring network topology is studied in the same way the 16-node ring network was studied in the previous section, i.e. in terms of blocking probability, total number of colors used, number of recolorings and simulation time, considering the Small-Bucket algorithm. For this network, four levels are considered, i.e. $d = 4$, since this number leads to the lowest possible number of colors and also to the lowest number of recolorings, as can be seen in section 4.2.

4.6.1. Blocking Probability and Simulation Time

In this subsection, the blocking probability, as well as the simulation time, for the 8-node ring network are computed and analyzed. Figure 4.12 represents the blocking probability as a function of the average offered load per node considering $C = 48$, 56 and 64 colors and $d = 4$. The blocking probabilities calculated using the First-fit are also represented for comparison purposes. For some values of the blocking probability in Fig. 4.12 the simulation times are also presented.

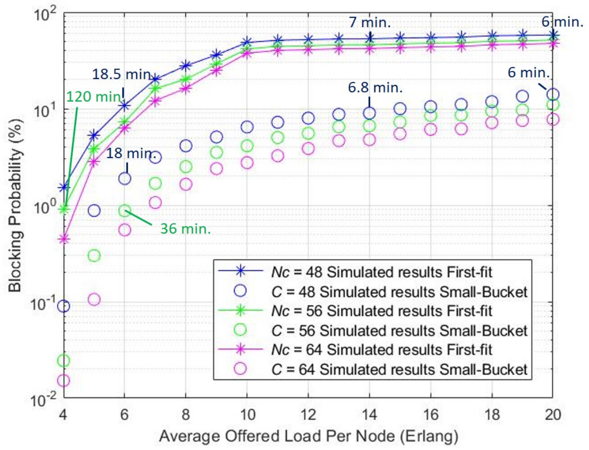


FIGURE 4.12. Blocking probability as a function of the average offered load per node for a bidirecctional 8-node ring network with $C = 48$, 56 and 64 colors per bucket considering the Small-Bucket as WA algorithm.

As can be seen from Figure 4.12, the Small-Bucket algorithm with C colors per bucket gives always lower blocking probability, compared to the First-fit with $N_c = C$. As already pointed out in the last section, this comparison is not completely fair, because the Small-Bucket algorithm is using a very high number of colors, when compared to the First-fit. Comparing Fig. 4.12 with Fig. 4.6 (for the 16-node ring network), it is observed that in the 8-node ring network, a 1% blocking probability is achieved with an average offered traffic per node of 4 E, whereas, in the 16-node ring network, a 0.9 E per node is required to reach the same probability, since the network dimension is larger in this last scenario.

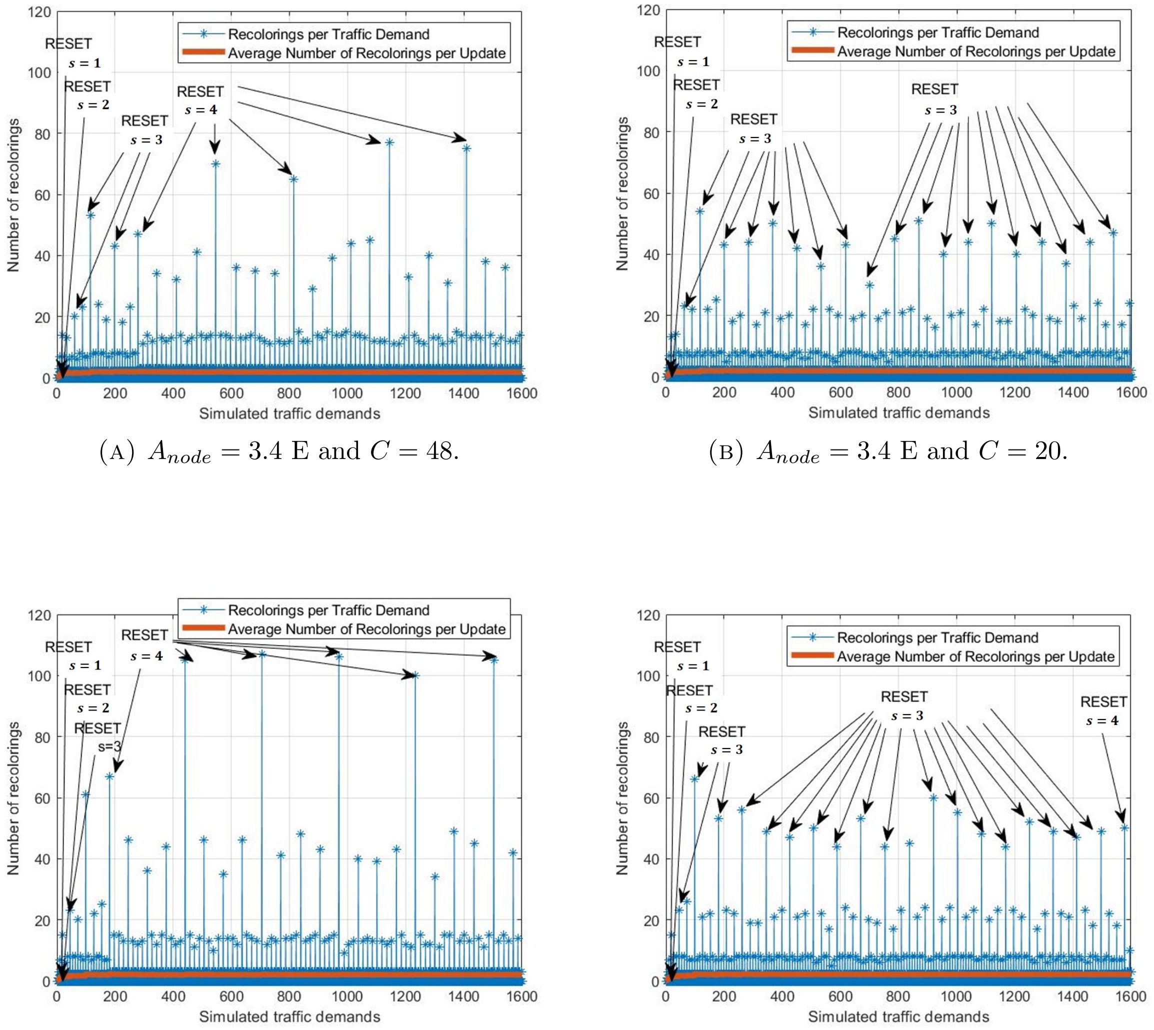
Once again, it is observed that, for the same network traffic, the simulation times obtained are very similar, although the Small-bucket reaches lower blocking probabilities.

For example, to achieve a blocking probability of 1% for $C = 56$, the Small-Bucket takes only 36 minutes ($A_{node} = 6$ E), while, the First-fit takes 120 minutes ($A_{node} = 4$ E).

4.6.2. Total Number of Colors Used and Number of Recolorings

In this subsection, the total number of colors and recolorings for the 8-node ring network are analyzed. The same total offered traffic conditions used for the 16 node ring network are going to be used for this 8 node ring network. So, the scenarios of 0.8 and 2 E traffic per node that correspond to a total traffic of, respectively, 96 and 240 E, for the 16-node ring network correspond to a 3.4 and 8.6 E traffic per node, for the 8-node ring network.

In Fig. 4.13 (A) through (D), the behavior of the number of recolorings as a function of the simulated traffic demands is studied, for $d = 4$, $A_{node} = 3.4$ E and 8.6 E. Fig. 4.13 (A) and (C) represent simulations with $C = 48$, while Fig. 4.13 (B) and (D) correspond to $C = 20$. The average number of recolorings is also represented in Figure 4.13.



(A) $A_{node} = 3.4$ E and $C = 48$.

(B) $A_{node} = 3.4$ E and $C = 20$.

(C) $A_{node} = 8.6$ E and $C = 48$.

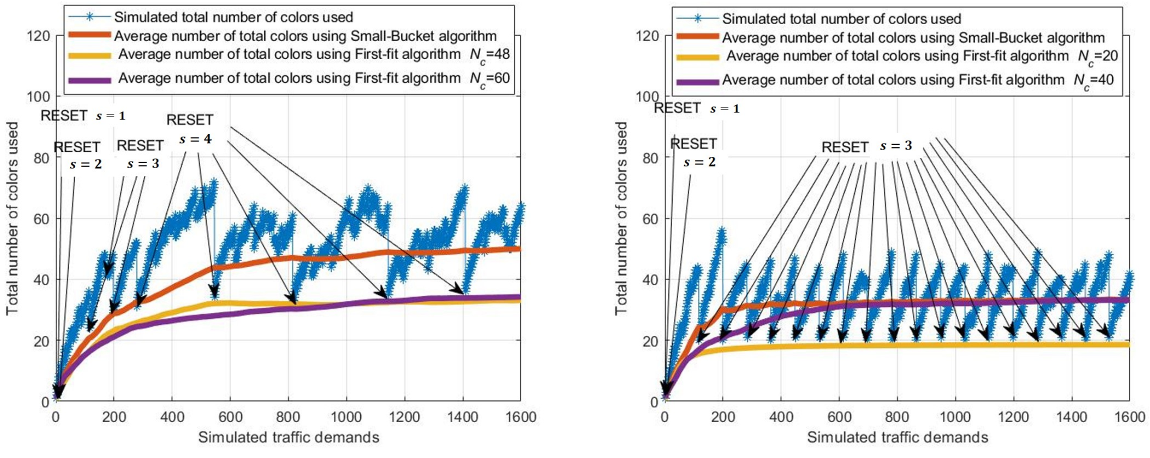
(D) $A_{node} = 8.6$ E and $C = 20$.

FIGURE 4.13. Simulated number of recolorings as a function of the simulated traffic demands and average offered load per node for the 8-node ring network with $d = 4$.

The analysis of Fig. 4.13 will be done by comparing its results with the ones of Fig. 4.9. Starting with Fig. 4.13 (A) and Fig. 4.9 (A), with $C = 48$, it can be observed that despite the number of big RESETS remains the same, the frequency of the number of recolorings in the lower floors increases, and the number of buckets per level (s) also increases. For example, in Fig 4.9 (A), the second floor has 2 recolorings between big RESETS, whereas in Fig. 4.13 (A), there are 3 recolorings. Also, the maximum number of buckets per level is $s = 3$ in Fig. 4.9 (A), whereas in Fig. 4.13 (A), this number increases to 4. These differences can be explained by noting that the higher frequency of recolorings in the lower floors causes big RESETs, involving multiple vertices than in Fig. 4.9 (A). These RESETs cause, as represented in Fig. 4.13 (A), the increase in the number of buckets per level to 4. The increase in the frequency of recolorings in the lower floors is due to the reduced number of links in each path relatively to the 16-node ring network, which allows the reuse of a greater number of colors, because the probability of two optical paths having links in common is lower.

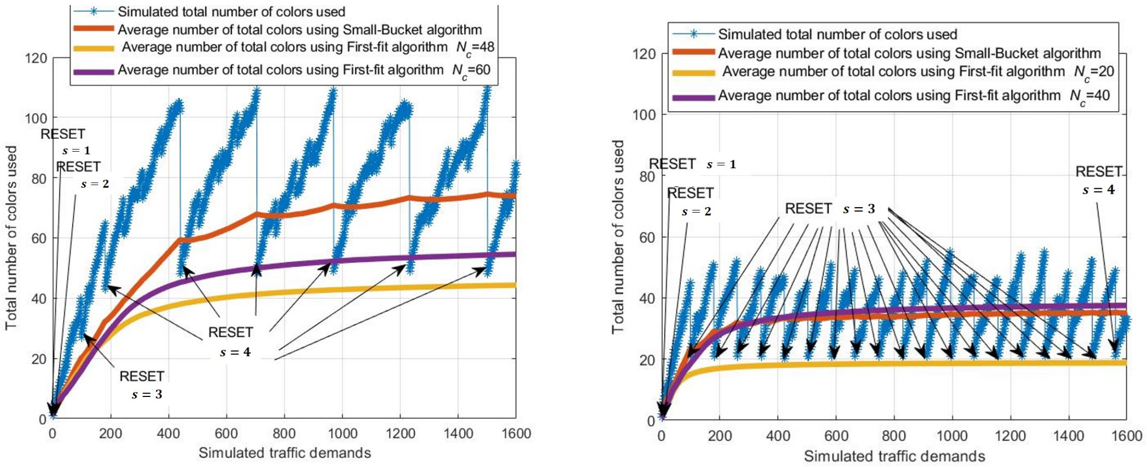
Comparing Fig. 4.9 (B) with Fig. 4.13 (B), with $C = 20$, it can observed that the number buckets per level remains the same, i.e., $s = 3$, but the number of big RESETS increases in Fig. 4.13 (B), as well as the frequency of the number of recolorings in the lower floors. In particular, we can observe that in Fig. 4.13 (B), there are 18 big RESETS, whereas in Fig. 9 (B) there are only 6. This behaviour can be explained by noting that the reduced capacity of the buckets causes a shorter time interval between recolorings in the RESET bucket (big RESETs). When the traffic per node increases, as in Fig. 4.13 (C), there are no differences in the behavior in comparison with Fig. 4.13 (A), except for the number of recolorings that is higher in Fig. 4.13 (C).

Fig. 4.14 shows the total number of colors used as a function of the simulated traffic demands, for the same parameters used in Fig. 4.13. The number of colors using the First-fit is also represented for comparison purposes with a maximum of $N_c = 48$ and 60 (Figs. 4.14 (A) and (C)) and $N_c = 20$ and 40 (Figs. 4.14 (B) and (D)) colors per bucket.



(A) $A_{node} = 3.4$ E and $C = 48$.

(B) $A_{node} = 3.4$ E and $C = 20$.



(C) $A_{node} = 8.6$ E and $C = 48$.

(D) $A_{node} = 8.6$ E and $C = 20$.

FIGURE 4.14. Total number of colors used as a function of the simulated traffic demands and average offered load per node for the 8-node ring network with $d = 4$.

It can be observed that with the increase of traffic offered (Figs. 4.14 (A) and (C)), the number of colors used also increases, because the buckets fill up more quickly. In general, comparing with the performance of the 16-node ring network, the average number of colors used by the Small-Bucket algorithm is similar, because the same average total network offered load is considered. For example, for $A_{node} = 3.4$ E and $C = 48$, the average number of colors used, approximately 50, is similar to the number obtained for the 16-node ring network for the same total traffic scenario ($A_{network} = 96$ E).

In Fig. 4.14, the total number of colors used considering the First-fit algorithm is also represented, for several values of N_c. It is observed in Fig. 4.14 (A) that the Small-Bucket with $C = 48$ has, on average, 20 more colors than the First-fit algorithm with $N_c = 48$ and 60, whereas in Fig. 4.14 (C), the difference is approximately 20 colors for $N_c = 60$, but increases to 30 for $N_c = 48$. On the other hand, the First-fit with $N_c = 40$ and the Small-Bucket lead to the same average number of colors used when the Small-Bucket with $C = 20$ is considered in both Fig. 4.14 (B) and (D).

In general, the difference between the number of colors used by the Small-Bucket and First-fit algorithms is smaller for the 8-node ring network (approximately 20 colors) compared to the difference of the number of colors observed for the 16-node ring network (approximately 25 colors), because the reduced number of levels (8-node ring network) implies a reduced number of buckets and buckets of lower capacity, which in practice leads to less color usage.

4.7. Small-Bucket Algorithm Results and Discussion for the UBN Network

The UBN network topology is studied in terms of blocking probability, total number of colors used, number of recolorings and simulation time, using the Small-Bucket algorithm for WA. For this network, six levels are considered, i.e. $d = 6$.

4.7.1. Blocking Probability and Simulation Time

In this subsection the blocking probability, as well as the simulation time, for the UBN network are computed and analyzed. Fig. 4.15 represents the blocking probability of the UBN network as a function of the average offered load per node considering $C = 48$, 56 and 64 colors in each bucket, and the Small-Bucket with $d = 6$ and the First-fit algorithm. For some values of the blocking probability in Fig. 4.15, the simulation times are also presented.

As can be seen from Figure 4.15, once again the Small-Bucket algorithm with C colors per bucket gives a lower blocking probability, compared to the First-fit with $N_c = C$. Once again, this comparison is not completely fair, because the Small-Bucket algorithm uses a very high total number of colors according to equation (4.1), in comparison with the First-fit. Comparing the First-fit algorithm with the Small-Bucket for the same network traffic, the simulation times obtained are very similar.

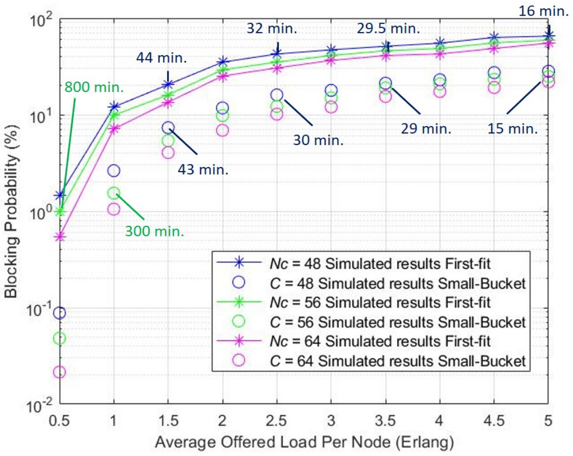
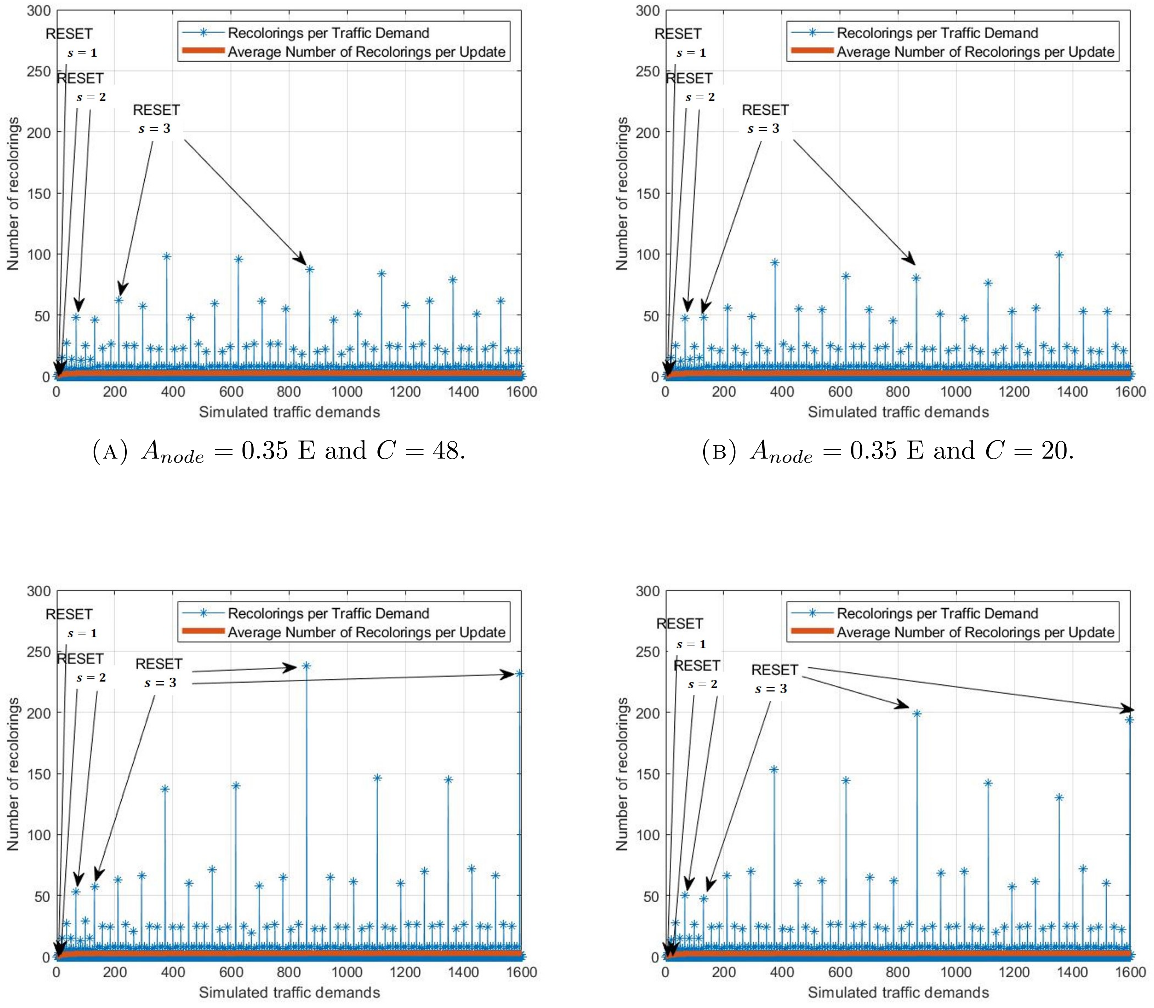


FIGURE 4.15. Blocking probability as a function of the average offered load per node for the UBN network with $C = 48$, 56 and 64 colors per bucket considering the Small-Bucket as WA algorithm.

4.7.2. Total Number of Colors Used and Number of Recolorings

In this subsection the total number of colors and the number of recolorings for the UBN network are computed and analyzed. The same total traffic offered to the 8 and 16-node ring networks is offered to the UBN network. The scenarios of 0.8 and 2 E traffic per node that correspond to a total traffic of, respectively, 96 and 240 E, for the 16-node ring network correspond to a 0.35 and 0.87 E traffic per node for the UBN network. In Fig. 4.16 (A) through (D), the number of recolorings as a function of the simulated traffic demands is studied. Fig. 4.16 (A), and (C) represent simulations with $C = 48$ colors, while Fig. 4.16 (B) and (D) correspond to $C = 20$. The average number of recolorings is also represented in Fig. 4.16.



(A) $A_{node} = 0.35$ E and $C = 48$.

(B) $A_{node} = 0.35$ E and $C = 20$.

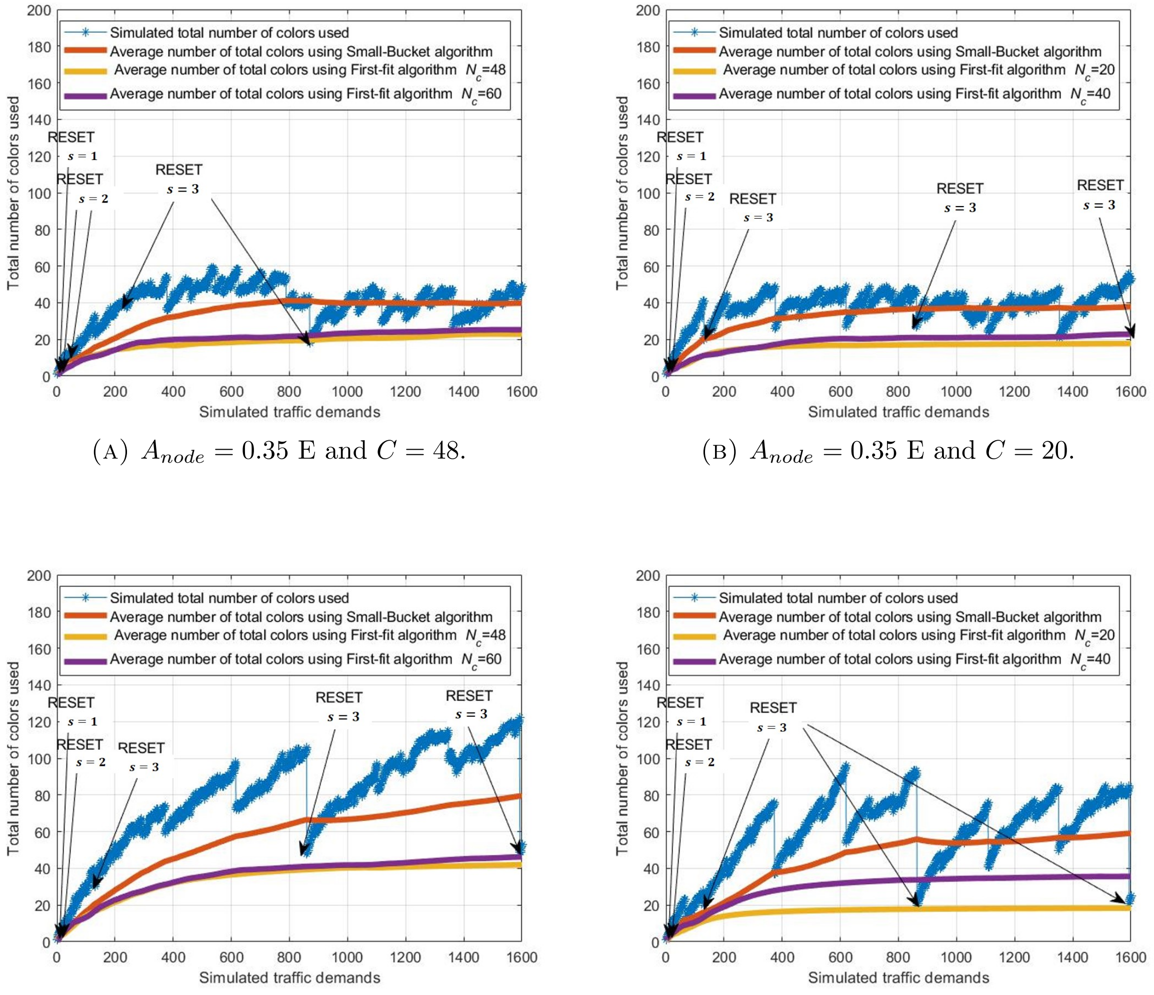
(C) $A_{node} = 0.87$ E and $C = 48$.

(D) $A_{node} = 0.87$ E and $C = 20$.

FIGURE 4.16. Simulated number of recolorings as a function of the simulated traffic demands and average offered load per node for the UBN network with $d = 6$.

Once again, the first 1600 demands are represented. The difference that is most visible in Fig. 4.16 in comparison with Figs. 4.13 and 4.9 is the decrease in the number of RESETs. RESETs are less frequent because, in this scenario, $d = 6$ levels are being considered compared, for example, to the 16-node ring network where $d = 5$. This additional level originates buckets with greater capacity and, thus, reduces the number of RESETs. For example, when the number of buckets per level is $s = 3$, the buckets of the last level ($i = 5$) have $C = 48$ available colors for coloring 243 possible demands ($243 = 3^5$), whereas in the 16-node ring network, the buckets on the last level ($i = 4$) have the same 48 available colors for coloring only 81 possible demands ($81 = 3^4$).

Fig. 4.17 shows the total number of colors used, as well as its average value, as a function of the simulated traffic demands, for the same parameters used in Fig. 4.16. The number of colors using the First-fit is also represented with a maximum of $N_c = 48$ and 60 (Figs. 4.17 (A) and (C)) and $N_c = 20$ and 40 (Figs. 4.17 (B) and (D)) colors.



(A) $A_{node} = 0.35$ E and $C = 48$.

(B) $A_{node} = 0.35$ E and $C = 20$.

(C) $A_{node} = 0.87$ E and $C = 48$.

(D) $A_{node} = 0.87$ E and $C = 20$.

FIGURE 4.17. Total number of colors used as a function of the simulated traffic demands and average offered load per node for the UBN network with $d = 6$.

In Fig. 4.17, it is observed that with the increase of offered traffic (0.35 E to 0.87 E), the number of colors used in the Small-Bucket also increases, because the buckets fill up more quickly as observed in Figs. 4.14 and 4.10. In general, comparing the performance of the Small-Bucket algorithm in the UBN network with the performance in the 16-node and 8-node ring networks, it is observed that the Small-Bucket algorithm uses a lower number of colors. This is due to the large storage capacity of buckets of the last level ($i = 5$), allowing the reuse of multiple colors. For the UBN network with $d = 6$, the buckets of the last level are able to assign $C = 48$ colors to $3^5 = 243$ vertices. For example, for $A_{node} = 0.35$ E and $C = 48$, the average number of colors used is approximately 40 colors whereas, under the same conditions, for both the 8 and 16-node ring networks, the average of colors used is approximately 50.

In Fig. 4.17, the total number of colors used considering the First-fit is also represented for various N_c. For this network and adopting the First-fit algorithm, it is observed that the variation of the maximum number of wavelengths per link (N_c) has no impact on the number of colors used, except in Fig. 4.17 (D) , for $A_{node} = 0.87$ E and $C = 20$. This is due to the fact that to color the 1600 traffic demands and, given the high number of distinct optical paths, there is no need to increase the average number of colors.

4.8. Conclusions

In this chapter, the Small-Bucket algorithm was described with detail and the simulator model was explained by resorting to a flowchart. The performance of the Small-Bucket algorithm was assessed for two ring networks and UBN network through the blocking probability, number of colors used, number of recolorings and simulation time. A comparison with the traditional First-fit algorithm was performed.

For the Small-Bucket, it is concluded, by considering the maximum number of colors/wavelengths per bucket (C) equal to the maximum number of wavelengths per link (N_c) in the First-fit, the blocking probability obtained using the Small-Bucket algorithm is significantly lower, because the Small-Bucket uses several buckets with capacity C, leading to a much higher average number of colors than the N_c colors of the First-fit. It was also observed that the blocking probability of a traffic demand in the Small-Bucket algorithm is independent of the number of levels considered.

The number of recolorings is related to the number of levels used. It has been shown that when the number of levels is high ($d = 6$, for the UBN), the frequency of RESETs occurrences is lower, because the system has a higher number of buckets and with larger capacity. This allows the distribution of demands over more color spaces (buckets), reducing the number of RESET occurrences. However, when RESETs occur, the number of recolorings is higher because it encompasses recolorings of vertices that were placed in buckets with increased capacity. On the other hand, for situations where the number of levels is low ($d = 4$, for the 8-node ring), the RESETs frequency is higher, due to the reduced number of buckets in the system and buckets of lower capacity.

Regarding the number of colors used, it is concluded that the number of colors used is similar for all the networks studied, because the same values of average offered load were chosen for all tested networks. In general, a greater number of colors is required when the Small-Bucket is considered as the WA algorithm, to obtain similar blocking probabilities as the ones given by the First-fit algorithm. This was observed for UBN and 16-node ring networks and it is due to the fact that the First-fit only considers a single coloring space, while the Small-Bucket algorithm considers different coloring spaces (buckets).

The number of recolorings computed and the number of colors used are the two major weaknesses of the Small-Bucket algorithm application to optical networks, because to face recolorings with a such higher number of colors, fast tunable transceivers would be required, leading to an increase of the network cost and management. In the Small-Bucket algorithm, a demand may not be blocked because there is a possibility of recoloring. In First-fit there is no such possibility. The advantage of the Small-Bucket algorithm is that it is prepared for recolorings, allowing more network dynamism.

CHAPTER 5

Conclusions and Future Work

In this chapter, the main conclusions of this work and possible suggestions for future work are presented.

5.1. Conclusions

This work focused on studying different graph coloring algorithms and their comparison with traditional algorithms for solving the RWA problem in dynamic optical networks. The goal of a dynamic RWA problem is to minimise the blocking probability of a traffic demand.

Chapter 2 focused on the introduction of the dynamic optical networks fundamentals. In particular, the motivation for dynamic optical networks has been presented, as well as the basic concepts of their planning. The RWA algorithms studied and implemented in this work were also presented: the routing algorithm Fixed-Alternate and the WA algorithms First-fit, Most-used, Random, Greedy and Small-Bucket.

Chapter 3 starts with the analysis of the blocking probability in a point-to-point scenario by using a $M/M/c/c$ model. This model was implemented and validated for several scenarios. Next, the blocking probability is studied in a dynamic optical network, for several RWA algorithms. In addition, the flowchart that describes the planning tool developed in this work to simulate RWA algorithms for dynamic optical networks has been also presented. Then, the developed simulator was tested and validated for different network topologies (8 and 16-node ring networks, NSFNET and British Telecom) and RWA algorithms (Fixed-Alternate Routing, First-fit, Most-Used, Random and Greedy) using a Monte-Carlo simulation with dynamic arrivals and departures of traffic demands, by comparison with literature results. The traffic simulations in the various tested networks allowed to evaluate the different WA algorithms, in terms of the blocking probability and simulation time. In general, the WA algorithms that produce lower blocking probabilities correspond to the First-fit and Greedy algorithms, while the Most-used and Random algorithms lead to more blocked demands. The First-fit requires a lower simulation time to achieve a target blocking probability, while the Greedy algorithm requires more time. Moreover, an analytical formalism based on the reduced load approximation was used to model a simple network scenario, and its results are in very good agreement to the ones obtained with the simulator developed.

Chapter 4 starts by explaining the Small-Bucket graph coloring technique as a WA algorithm, and its implementation through a flowchart. The blocking probability, simulation time, number of recolorings and number of colors used by the Small-Bucket algorithm for UBN and ring networks of 8 and 16 nodes has been assessed, and a comparison with the First-fit algorithm has been performed.

It was observed that the Small-Bucket algorithm applied to optical networks tends to lead to much lower blocking probabilities than the First-fit, but in contrast, it requires more colors and recolorings. For example, for the 16-node ring network and a maximum of 48 colors per bucket, the blocking probability for an average traffic load per node of 2 E is 10%, with an average of 75 colors used and 2.77 average recolorings per update. The number of recolorings can be much higher than this value, especially when a RESET occurs. Using the First-fit algorithm, the simulated blocking probability is 30% with a total of 48 colors used. Despite the advantage of leading to lower blocking probabilities, the Small-Bucket algorithm presents as major disadvantages the high number of colors used and the number of required wavelength reconfigurations, which may prevent its applicability in optical networks, but it offers greater flexibility in network planning. Due to the high number of recolorings, this algorithm will bring more benefits when applied to metropolitan or access networks.

5.2. Future Work

In the following, some proposals of future work are presented:

- Adapt the Small-Bucket algorithm in order to minimize the number of recolorings per network update, while leading to similar target blocking probabilities. For example, by allowing the use of the same color in different buckets, if the corresponding lightpaths have not common links;
- Extend the network analysis to other dynamic traffic patterns, besides the uniform case analyzed in this work, where all the nodes offer the same average traffic to the network. Consider, for example, a situation where two or three nodes are responsible for generating almost all the network traffic [19];
- Detailed study of the Big-Bucket algorithm as a graph coloring technique for WA in dynamic optical networks [3].

References

[1] J. Simmons, *Optical Network Design and Planning*, 2nd edition, Eds. Holmdel: Springer, 2014, pp. 349-395.

[2] P. Rajalakshmi, and A. Jhunjhunwala, “Re-Routing at Critical Nodes to Enhance Performance of Wavelength Reassignment in All-Optical WDM Networks Without Wavelength Conversion,” in *Journal of Lightwave Technology*, vol. 26, no. 17, pp. 3021-3029 September 2008.

[3] L. Barba et al., “Dynamic Graph Coloring,” *Algorithmica*, vol. 81, no. 4, pp. 1319-1341, June 2018.

[4] Graph coloring techniques for planning dynamic optical networks - video demonstration and software tool, 2021. [Online]. Available: https://drive.google.com/drive/folders/1sCK66hUvYOZcX-PCGRtU7i8KXWml1ek0?usp=sharing .

[5] B. Vieira, D. Conceição, J. Pedro, and J. Pires, “Evaluating the Impact of Physical Impairments on the Blocking Performance of an All-optical WDM Ring under Dynamic Traffic,” *Journal of Optical Communications*, vol. 28, no. 1, pp. 66-72, Nov. 2007.

[6] X. Chu, B. Li and I. Chlamtac, “Wavelength Converter Placement Under Different RWA Algorithms in Wavelength-Routed All-Optical Networks,” *IEEE Transactions on Communications*, vol. 51, no. 4, pp. 607-617, Apr. 2003.

[7] N. Benzaoui, M. Szczerban Gonzalez, J. Manuel Estarán , H. Mardoyan ,W. Lautenschlaeger, U. Gebhard, L. Dembeck, S. Bigo, and Y. Pointurier, “Deterministic Dynamic Networks,” *Journal of Lightwave Technology*, vol. 37, no. 14, pp. 3465-3474, July 2019.

[8] G. Lewis, *Cloud Computing*, Eds. Long Beach: 2017, pp. 8-9.

[9] L. Yuan, L. Qin, X. Lin , L. Chang, and W. Zhang, “Effective and Efficient Dynamic Graph Coloring,” *Proc. VLDB Endow. 11*, vol. 11, no. 3, pp. 338-351, Nov. 2017.

[10] D. Gross et al, *Fundamentals of Queueing Theory*, 4th edition. Hoboken: Wiley, 2008.

[11] Ng. Chee-Hock, and Soong Boon-Hee, *Queueing Modelling Fundamentals: with Applications in Communication Networks*, 2nd edition, Eds. Hoboken: Wiley, 2008, pp. 103-140.

[12] M. Jeruchim, P. Balaban, and K. Shanmugan, “Monte Carlo Simulation and Generation of Random Numbers,” in *Simulation of Communication Systems: Modeling, Methodology, and Techniques*, Eds. New York: Kluwer, 2000, pp. 371-406.

[13] Slides from the course in Optical Networks in *Telecommunications and Computer Engineering MSc.* ISCTE-IUL, Lisbon, Portugal, 2019.

[14] K. Ross, “The Reduced Load Approximation for Single-Service Networks,” in *Multiservice Loss Models for Broadband Telecommunication Networks*, Eds. Philadelphia: Springer, 1995, pp. 181-187.

[15] I. Duarte, “Exploring Graph Coloring Heuristics for Optical Networks Planning”, Master dissertation in Telecommunications and Computer Engineering. Lisbon, ISCTE-IUL, Oct. 2020.

[16] D. J. Ives, A. Lord, P. Wright, and S. J. Savory, “Quantifying the Impact of Non-

-linear Impairments on Blocking Load in Elastic Optical Networks," in *Optical Fiber Communication Conference 2014 ©OSA*, San Francisco, 2014, paper W2A.55.

[17] R. M. R. Lewis, *A Guide to Graph Coloring.* Switzerland: Springer, 2016.

[18] L. Danh et al., "AgileDCN: An Agile Reconfigurable Optical Data Center Network Architecture," *Journal of Lightwave Technology*, vol. 38, no. 18, pp. 4922-4934, June 2020.

[19] B. Correia et al., "Power Control Strategies and Network Performance Assessment for C+L+S Multiband Optical Transport," *Journal of Optical Communications and Networking*, vol. 13, no. 7, pp. 147-157, April 2021.

[20] X. Sun, Y. Li, I. Lambadaris, and Y. Q. Zhao, "Performance Analysis of First-fit Wavelength Assignment Algorithm in Optical Networks," in *Proceedings of the 7th International Conference on Telecommunications*, Zagreb, 2003, pp. 403-409.

[21] M. Niksirat, S. Hashemi, and M. Ghatee, "Branch-and-price Algorithm for Fuzzy Integer Programming Problems with Block Angular Structure," *Fuzzy Sets Syst.*, vol. 296, no. 2, pp. 70-96, Oct. 2016.

[22] T. LaQuey, "NSFNET," in *The User's Directory of Computer Networks*, Eds. Boston: Digital Press, 1990, pp. 247-250.

[23] "CONUS WDM Network Topology," Monarch Network Architects, 2012. [Online]. Available: http://monarchna.com/topology.html.

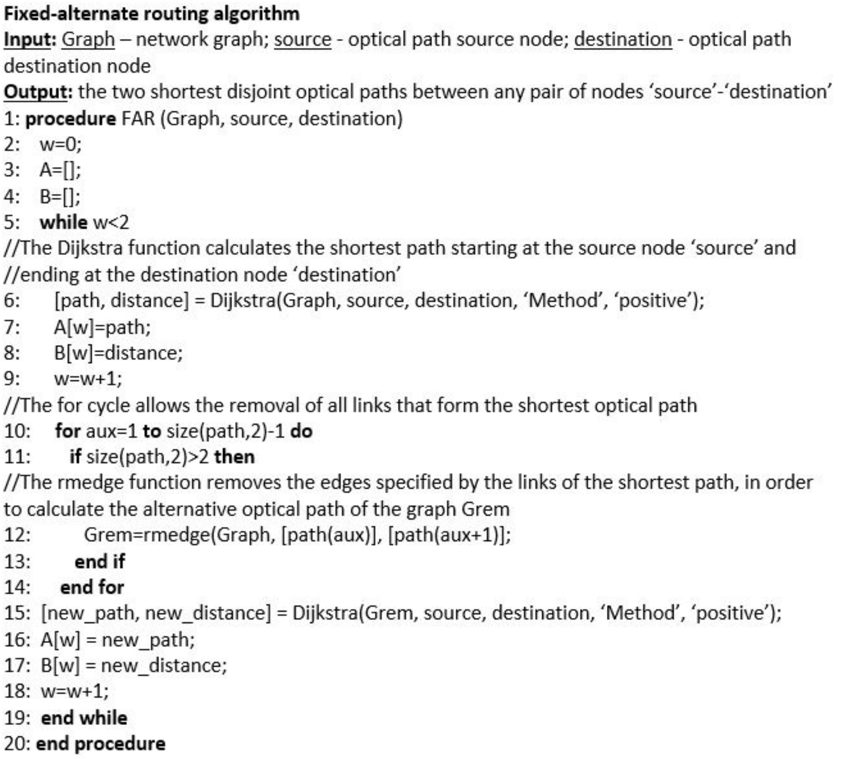
[24] R. Godoi, P. Hernandez, and E. Luque, "Control Evaluation in a LVoD System Based on a Peer-to-Peer Multicast Scheme," *Journal of Computer Science and Technology*, vol. 8, no. 2, pp. 97-103, July 2008.

[25] S. Ramamurthy and B. Mukherjee, "Fixed-Alternate Routing and Wavelength Conversion in Wavelength-routed Optical Networks," in *IEEE GLOBECOM 1998*, Sydney, 1998, pp. 2295-2302.

APPENDIX A

Pseudocodes of the Implemented Algorithms

In the development of the dynamic optical network simulator, several routing and WA algorithms have to be developed. This appendix presents the pseudocodes of these algorithms. The algorithm considered to carry out traffic routing corresponds to the Fixed-Alternate Routing, whose pseudocode is shown in Figure A.1. The procedure is performed before the network goes into operation, aiming to calculate the two shortest disjoint optical paths between any pair of nodes.



Fixed-alternate routing algorithm

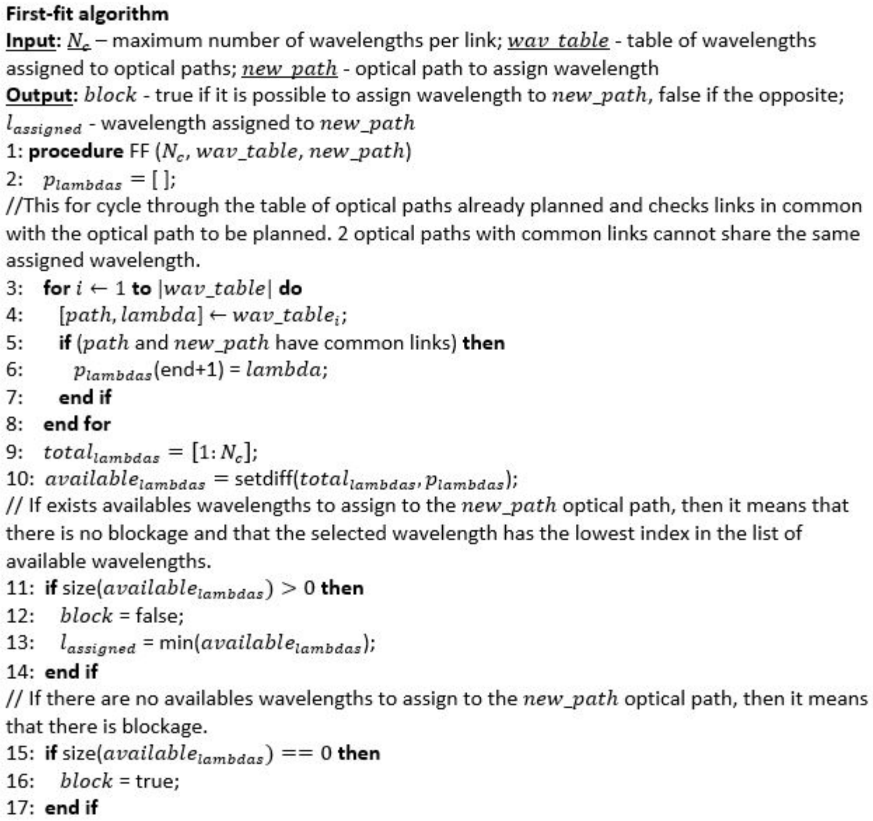
Input: Graph – network graph; source - optical path source node; destination - optical path destination node

Output: the two shortest disjoint optical paths between any pair of nodes 'source'-'destination'

```
procedure FAR (Graph, source, destination)
  w=0;
  A=[];
  B=[];
  while w<2
//The Dijkstra function calculates the shortest path starting at the source node 'source' and
//ending at the destination node 'destination'
     [path, distance] = Dijkstra(Graph, source, destination, 'Method', 'positive');
     A[w]=path;
     B[w]=distance;
     w=w+1;
//The for cycle allows the removal of all links that form the shortest optical path
     for aux=1 to size(path,2)-1 do
        if size(path,2)>2 then
//The rmedge function removes the edges specified by the links of the shortest path, in order
to calculate the alternative optical path of the graph Grem
           Grem=rmedge(Graph, [path(aux)], [path(aux+1)];
        end if
     end for
  [new_path, new_distance] = Dijkstra(Grem, source, destination, 'Method', 'positive');
  A[w] = new_path;
  B[w] = new_distance;
  w=w+1;
  end while
end procedure
```

Figure A.1. Fixed-Alternate Routing - Pseudocode.

To solve the problem of wavelength assignment, several algorithms were implemented, as the First-fit, Most-used, Random, Greedy graph coloring strategy and Small-Bucket. In the First-fit algorithm, all available wavelengths are indexed and the available wavelength with the lowest index is chosen. In the Most-used algorithm, a higher priority is given to the wavelengths assigned to more links. The pseudocode of the First-fit algorithm is presented in Figure A.2.

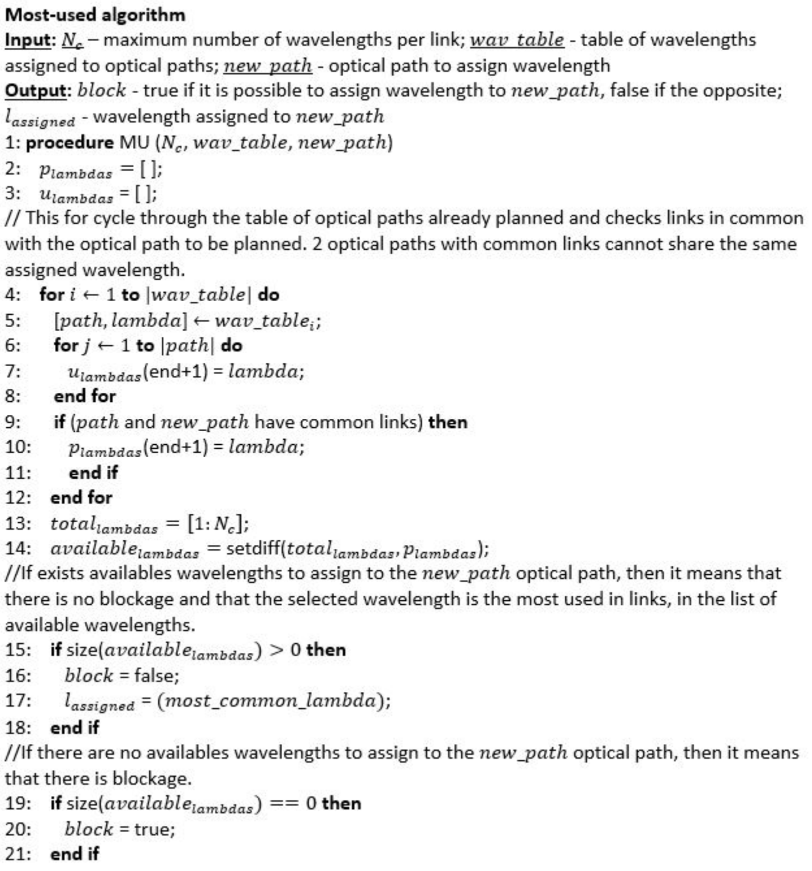


```
First-fit algorithm
Input: $N_c$ – maximum number of wavelengths per link; $wav\_table$ - table of wavelengths
assigned to optical paths; $new\_path$ - optical path to assign wavelength
Output: $block$ - true if it is possible to assign wavelength to $new\_path$, false if the opposite;
$l_{assigned}$ - wavelength assigned to $new\_path$
procedure FF ($N_c$, $wav\_table$, $new\_path$)
$p_{lambdas}$ = [ ];
//This for cycle through the table of optical paths already planned and checks links in common
with the optical path to be planned. 2 optical paths with common links cannot share the same
assigned wavelength.
for $i \leftarrow 1$ to $|wav\_table|$ do
    $[path, lambda] \leftarrow wav\_table_i$;
    if ($path$ and $new\_path$ have common links) then
        $p_{lambdas}$(end+1) = $lambda$;
    end if
end for
$total_{lambdas} = [1{:}N_c]$;
$available_{lambdas}$ = setdiff($total_{lambdas}, p_{lambdas}$);
// If exists availables wavelengths to assign to the $new\_path$ optical path, then it means that
there is no blockage and that the selected wavelength has the lowest index in the list of
available wavelengths.
if size($available_{lambdas}$) > 0 then
    $block$ = false;
    $l_{assigned}$ = min($available_{lambdas}$);
end if
// If there are no availables wavelengths to assign to the $new\_path$ optical path, then it means
that there is blockage.
if size($available_{lambdas}$) == 0 then
    $block$ = true;
end if
```

Figure A.2. First-fit algorithm - Pseudocode.

The purpose of the developed First-fit algorithm is to calculate the wavelength, designed $l_{assigned}$, for the optical path, new_{path}. Through the assignments of previous wavelengths, wav_{table}, it is possible to apply the First-fit algorithm, with the objective that the available wavelength with the lowest index is chosen. If the index of the chosen wavelength is higher than N_c (maximum number of wavelengths per link), there is a block in the WA for the new_{path}.

The pseudocode of the Most-used algorithm is shown in Figure A.3.



Most-used algorithm

Input: N_c – maximum number of wavelengths per link; *wav table* - table of wavelengths assigned to optical paths; *new path* - optical path to assign wavelength

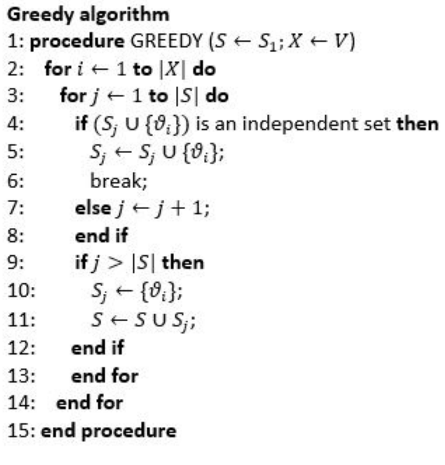
Output: *block* - true if it is possible to assign wavelength to *new_path*, false if the opposite; $l_{assigned}$ - wavelength assigned to *new_path*

```
procedure MU ($N_c$, $wav\_table$, $new\_path$)
  $p_{lambdas}$ = [ ];
  $u_{lambdas}$ = [ ];
// This for cycle through the table of optical paths already planned and checks links in common with the optical path to be planned. 2 optical paths with common links cannot share the same assigned wavelength.
  for $i \leftarrow 1$ to $|wav\_table|$ do
    [$path, lambda$] $\leftarrow wav\_table_i$;
    for $j \leftarrow 1$ to $|path|$ do
      $u_{lambdas}$(end+1) = $lambda$;
    end for
    if ($path$ and $new\_path$ have common links) then
      $p_{lambdas}$(end+1) = $lambda$;
    end if
  end for
  $total_{lambdas} = [1{:}N_c]$;
  $available_{lambdas}$ = setdiff($total_{lambdas}, p_{lambdas}$);
//If exists availables wavelengths to assign to the $new\_path$ optical path, then it means that there is no blockage and that the selected wavelength is the most used in links, in the list of available wavelengths.
  if size($available_{lambdas}$) > 0 then
    $block$ = false;
    $l_{assigned}$ = ($most\_common\_lambda$);
  end if
//If there are no availables wavelengths to assign to the $new\_path$ optical path, then it means that there is blockage.
  if size($available_{lambdas}$) == 0 then
    $block$ = true;
  end if
```

Figure A.3. Most-used algorithm - Pseudocode.

The idea of the Most-used algorithm is identical to that of the First-fit, except that there is a variable, $u_{lambdas}$, which stores the number of times each wavelength has been assigned per link. To the wavelength most times assigned is given a higher priority to the next wavelength attribution, $lambda_{assigned}$. If the conditions are not met, the second wavelength with more links assigned is attempted, and so on.

The pseudocode of the Greedy graph coloring strategy is shown in Figure A.4.



Greedy algorithm

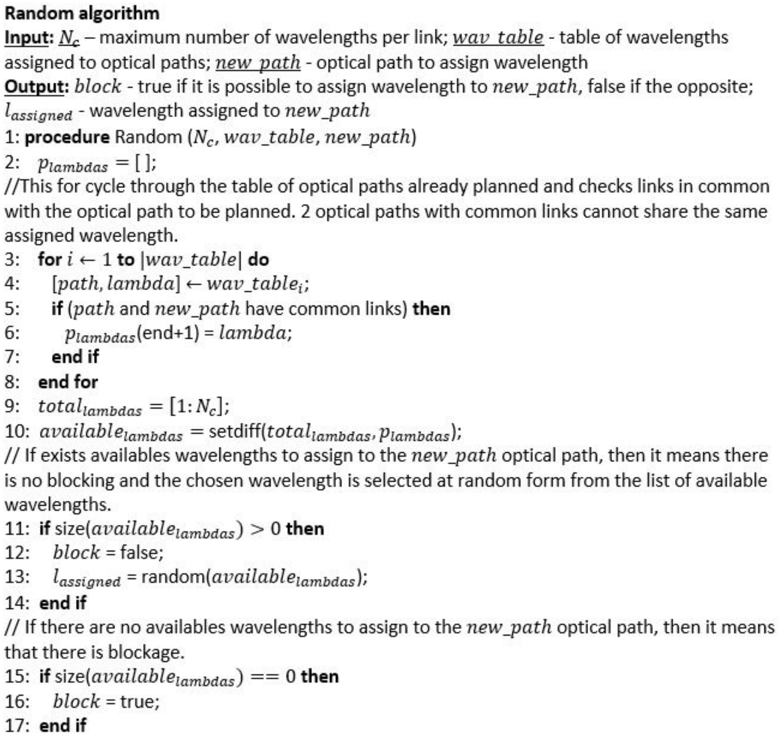
```
procedure GREEDY ($S \leftarrow S_1; X \leftarrow V$)
  for $i \leftarrow 1$ to $|X|$ do
    for $j \leftarrow 1$ to $|S|$ do
      if ($S_j \cup \{\vartheta_i\}$) is an independent set then
        $S_j \leftarrow S_j \cup \{\vartheta_i\}$;
        break;
      else $j \leftarrow j + 1$;
      end if
      if $j > |S|$ then
        $S_j \leftarrow \{\vartheta_i\}$;
        $S \leftarrow S \cup S_j$;
      end if
    end for
  end for
end procedure
```

Figure A.4. Greedy graph coloring strategy - Pseudocode.

The Greedy graph coloring strategy was implemented following [17]. In Figure A.4, the pseudocode of the implemented algorithm is described. For the set of all vertices/paths not initially assigned to a color (X) is assigned the set of all vertices in order of arrival (V). Subsequently, the set S, is the set that represents the colors associated with the vertices of X, with maximum capacity equal to the maximum number of wavelengths per link. Initially, this set is filled with the color S_1, since there is already the first color to be assigned to a vertex.

Then, the set of vertices of X is traversed (v_i), to check if vertex v_i is possible to assign a color S_j, of the set S. If ($S_j \cup v_i$) is an independent set, then the color S_j was not assigned to an adjacent vertex of v_i and, therefore, the color S_j is assigned to the vertex v_i. If ($S_j \cup v_i$) is not an independent set, then S_j has been assigned to an adjacent vertex of v_i and, therefore, a new color is attempted (the value of j is incremented). When all the colors of S have already been tested, a new color is created and the color is assigned to the vertex/path v_i, as well as its addition to S. If the maximum dimension of the set S is reached (*wav-per-link*), it means that the traffic demand is blocked.

The pseudocode of the Random algorithm is shown in Figure A.5.



```
Random algorithm
Input: $N_c$ – maximum number of wavelengths per link; wav_table - table of wavelengths
assigned to optical paths; new_path - optical path to assign wavelength
Output: block - true if it is possible to assign wavelength to new_path, false if the opposite;
$l_{assigned}$ - wavelength assigned to new_path
procedure Random ($N_c$, wav_table, new_path)
    $p_{lambdas}$ = [ ];
    //This for cycle through the table of optical paths already planned and checks links in common
    with the optical path to be planned. 2 optical paths with common links cannot share the same
    assigned wavelength.
    for $i \leftarrow 1$ to |wav_table| do
        [path, lambda] ← $wav\_table_i$;
        if (path and new_path have common links) then
            $p_{lambdas}$(end+1) = lambda;
        end if
    end for
    $total_{lambdas}$ = [1: $N_c$];
    $available_{lambdas}$ = setdiff($total_{lambdas}$, $p_{lambdas}$);
    // If exists availables wavelengths to assign to the new_path optical path, then it means there
    is no blocking and the chosen wavelength is selected at random form from the list of available
    wavelengths.
    if size($available_{lambdas}$) > 0 then
        block = false;
        $l_{assigned}$ = random($available_{lambdas}$);
    end if
    // If there are no availables wavelengths to assign to the new_path optical path, then it means
    that there is blockage.
    if size($available_{lambdas}$) == 0 then
        block = true;
    end if
```

Figure A.5. Random algorithm - Pseudocode.

The implementation of the Random algorithm is very similar to the implementation of the First-fit and Most-used algorithms. Initially, the algorithm checks which wavelengths are available to allocate to the new traffic demand ($available_{lambdas}$), bearing in mind that traffic demands with optical links in common cannot share the same wavelength.

At the end of the procedure, a distinct part of the First-fit and Most-used algorithms, a wavelength is randomly selected from $available_{lambdas}$ list.

In order to validate the implementation of the First-fit, Most-used, Random and Greedy pseudocodes, for different static optical networks topologies, the total number of assigned wavelengths obtained was calculated, for comparison with [15]. Table A.1 shows the total number of assigned wavelengths obtained by the First-fit, Most-used, Random and Greedy algorithms, for the networks presented in Figure 3.15 (NSFNET-14), Figure B.1 (Ring-5 nodes), Figure B.2 (Ring-10 nodes) and Figure B.3 (COST-239). In order to validate the WA algorithms with results already known in static networks, for the execution of each WA algorithms is passed as a function parameter, a table whose lines represents the various possible optical paths of the network and the columns represents the distances and the wavelength assigned. In contrast to the dynamic scenario, in a static scenario, there are several path demands to assign wavelength at the same time, and before using any algorithm for assigning wavelengths, the traffic demands are first ordered. In this case, before the execution of the WA algorithms, traffic demands with the shortest path appear first in the list of assignment of wavelengths (Shortest Path First ordering strategy).

Table A.1. Total number of assigned wavelengths obtained by First-fit, Most-used, Random and Greedy algorithms for the NSFNET-14, Ring-5 nodes, Ring-10 nodes and COST-239 networks.

Networks	First-fit	Most-used	Greedy	Random
Ring-5 nodes	4	4	3	5
Ring-10 nodes	16	16	15	23
COST-239	8	8	8	11
NSFNET-14	16	15	14	20

With these results, the implemented code for each of the conventional WA algorithms is validated, because the simulated and presented values in Table A.1 are very similar to the values of [15]. From this validation, the implemented algorithms were adapted to dynamic optical networks.

APPENDIX B

Network Topologies

The physical topology of the networks used in Chapter 3 is the following. Note that the nodes are numbered sequentially with no particular order.

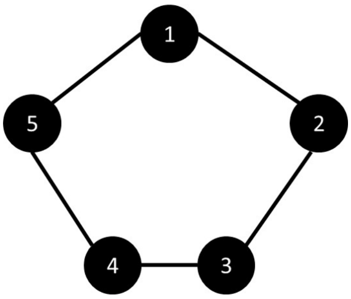


Figure B.1. Ring network with 5 nodes.

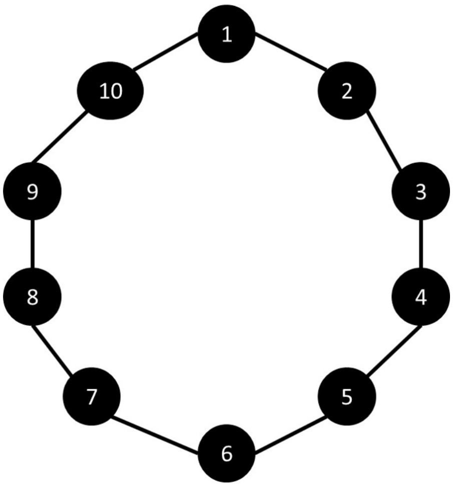


Figure B.2. Ring network with 10 nodes.

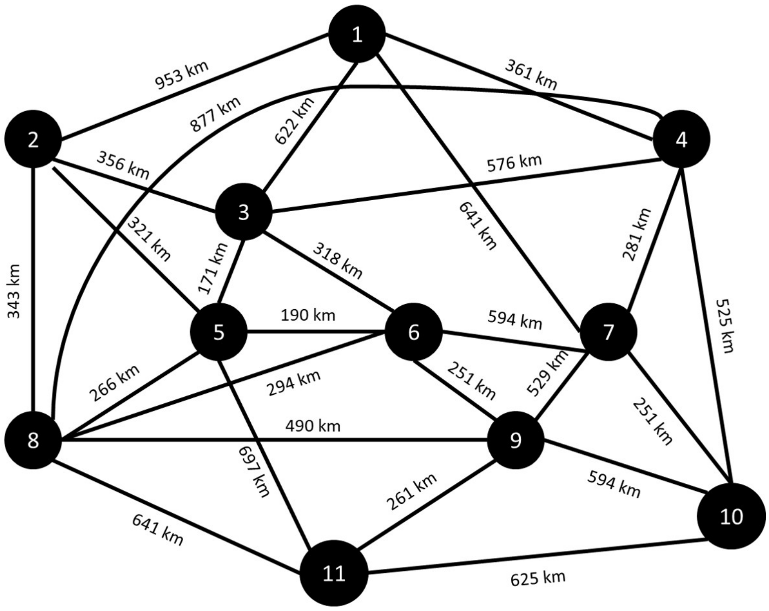


FIGURE B.3. COST-239 network.

Performance Analysis of a Graph Coloring Algorithm for Wavelength Assignment in Dynamic Optical Networks

Pedro Fonseca
Information Science and Technology
ISCTE-IUL
Lisbon, Portugal
paffa@iscte-iul.pt

Luís Cancela
Optical Communications and Photonics
Instituto de Telecomunicações
ISCTE-IUL
Lisbon, Portugal
luis.cancela@iscte-iul.pt

João Rebola
Optical Communications and Photonics
Instituto de Telecomunicações
ISCTE-IUL
Lisbon, Portugal
joao.rebola@iscte-iul.pt

***Abstract*—Dynamic optical networks will be crucial in global optical communications in the next 5-10 years. Efficient network planning tools that deal with Routing and Wavelength Assignment (RWA) problems are of paramount relevance in this dynamic scenario. In this work, a simulator for planning dynamic optical networks was developed, and several real networks were tested. In this simulator, we have implemented a graph coloring wavelength assignment (WA) algorithm named Small-Buckets algorithm that allows recoloring to occur. A comparison performance with the First-fit algorithm is performed in terms of the blocking probability, number of recolorings, number of colors used and simulation time. It is concluded that the Small-Buckets algorithm originates lower blocking probabilities than the ones obtained with the First-fit algorithm. However, to reach these low blocking probabilities, the Small-Buckets algorithm makes use of a larger number of wavelengths and recolorings.**

***Index Terms*—Dynamic Optical Networks, Graph Coloring, Routing and Wavelength Assignment, Small-Buckets algorithm**

I. Introduction

Optical transport networks are usually quasi-static, in the sense that connections often remain in service for a long period of time [1]. Nevertheless, with the need for on-demand services, fuelled by applications such as cloud computing and grid computing, together with the availability of an increasingly dynamic network infrastructure, it is expected that today's quasi-static optical networks turn into dynamic optical networks in the next 5-10 years [1].

In the optical network layer, Routing and Wavelength Assignment (RWA) are fundamental functions to transport data in an efficient way. For dynamic optical networks, the heuristics used for wavelength assignment (WA) in static networks, like First-fit, Most-used and Random, can also be used but, in this case, the sorting strategy is not necessary [2]. In a dynamic environment, every time a traffic demand arrives, a new wavelength must be found for routing the respective demand, without changing the wavelengths already in use in the network. This scenario can lead to a high blocking probability, when innumerous almost simultaneous traffic demands arrive, and the network is almost at its peak traffic load. In this case of network congestion, the network performance can be improved by letting wavelength reconfigurations to occur [3], in order to reduce the blocking probability.

Graph Coloring algorithms can be applied in many different areas, like optical networking, social networking, chemistry, scheduling and computer networking. The problem of Graph Coloring consists in coloring all the graph vertices with the minimum number of colors so that no vertices connected by an edge are given the same color [4].

To the best of our knowledge, graph coloring techniques for WA in optical dynamic networks have not been studied yet. A simple graph coloring WA algorithm for this scenario could be based on running the Greedy algorithm every time a new traffic demand arrives, which implies that all the vertices should be recolored. Another algorithm example is using a technique that just colors the demand that has arrived, which would be equivalent using the First-fit algorithm. In this work, we implement and analyze a graph coloring algorithm for the dynamic scenario that does not require the recoloring of all the vertices. This algorithm is named Small-Buckets algorithm and was initially proposed in [5]. In the Small-Buckets algorithm, the traffic demands are distributed into a set of buckets, each bucket with its own set of wavelengths (colors). A comparison with the First-fit algorithm is performed, in terms of the number of wavelengths used, blocking probability and simulation time. The number of recolorings is also studied.

The remainder of the paper is organized as follows. In Section II, the Small-Buckets algorithm is explained. In Section III, the developed RWA planning tool is explained and the network topologies studied are presented. In Section IV, the results are discussed in terms of blocking probability of a traffic demand, number of colors used, number of recolorings and simulation time. Also, a validation and a comparative performance study with the First-fit algorithm is done in this section. Finally, the conclusions are presented in Section V.

II. Small-Buckets Algorithm

In this section, the Small-Buckets is explained and an small example of its operation is presented.

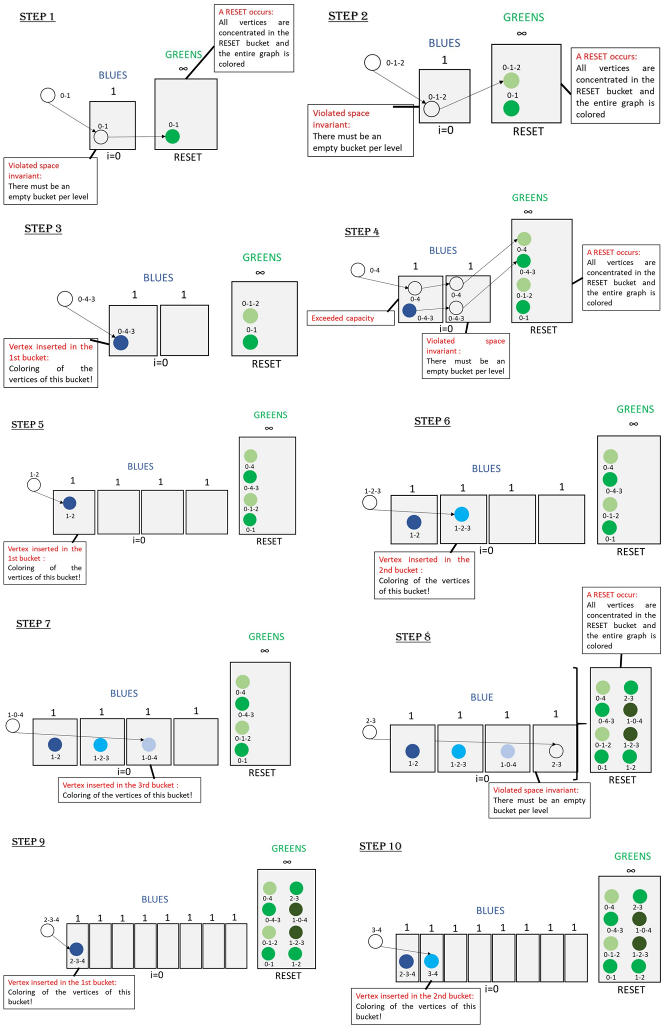
The Small-Buckets is a dynamic graph coloring algorithm [5] that is applied, in this work, for WA in dynamic networks. In this algorithm, a set of buckets is defined, where inside each bucket, several traffic demands can be accommodated. Each bucket has its own set of colors, so that two adjacent vertices do not have the same color. As a rule, there is a sequence of buckets of increasing size, organized in d levels, each containing s buckets, and a RESET bucket. The RESET bucket is a special bucket, where all demands are placed whenever all the levels are unable to accommodate a new demand. This bucket has an infinite capacity for vertices (i.e. demands), but like the other buckets, it has a limited number of colors.

The idea of the algorithm corresponds to the placement of a new demand in a bucket at level $i = 0$. If this is the last empty bucket at that level, placing it violates the main rule of the algorithm: per level there must be at least one empty bucket. Considering that the level $i = 0$ does not have available buckets, all vertices are shifted to the first empty bucket of the next level and then the WA using the specific color set of this new bucket is attempted. These shifts are repeated whenever the main rule is not fulfilled, up to a limit situation where all vertices are shifted to and recolored in the RESET bucket. When these level changes occur, vertex recoloring must also occur. In optical networks, a recoloring means that a wavelength reconfiguration needs to be performed. The number of recolorings is an important performance metric since it is highly related to the blocking probability. More recolorings mean a decrease in the blocking probability, but it also means that more colors are used.

Whenever there are recolorings it is important to check if traffic demands are blocked. In these recolorings, if there is no wavelength available for an already served traffic demand, that traffic demand is blocked, which is a situation that should not happen in a real situation. For departures of traffic demands, the bucket where the vertex of the corresponding path is located is searched. Once found, that path is removed from the bucket.

Two limit scenarios can be considered. The first one corresponds to perform the maximum number of recolorings i.e., N recolorings per update. This is possible if, for example, the Greedy algorithm is used for each demand arrival. The second scenario corresponds to perform the minimum number of recolorings, i.e., no recolorings. In this situation, a new color is assigned for each new demand and, so, the total number of colors increases with the number of demands. The Small-Buckets is between these two limit scenarios.

The Small-Buckets algorithm will be described next through an example, considering a ring topology with 5 nodes. In this network, it is possible to have a maximum of $N = 20$ distinct paths, considering a full mesh logical topology. The number of levels considered is $d = 1$ since the algorithm uses $d \times s$ buckets, grouped into d levels of s buckets each. All buckets at level i have capacity for s^i vertices. The RESET bucket has a maximum number of colors equal to C. In this example, blue colors are used for level $i = 0$ and green colors are assumed for the RESET bucket. Figure 1 represents the example of the Small-Buckets algorithm for WA using the initial configuration represented in the first step of Figure 1 and considering the first ten traffic demands corresponding to the ten steps of Figure 1.



A RESET occurs: All vertices are concentrated in the RESET bucket and the entire graph is colored

Violated space invariant: There must be an empty bucket per level

Vertex inserted in the 1st bucket: Coloring of the vertices of this bucket!

Exceeded capacity

Vertex inserted in the 2nd bucket : Coloring of the vertices of this bucket!

Vertex inserted in the 3rd bucket : Coloring of the vertices of this bucket!

A RESET occur: All vertices are concentrated in the RESET bucket and the entire graph is colored

Fig. 1. Small-Buckets algorithm example considering a 5 node ring topology with a full mesh logical topology. The first ten traffic demands are represented.

In the traffic demands represented in Fig. 1, there is a violation of the main rule in steps 1, 2, 4 and 8. In the remaining, it is possible to place the new vertex in the first empty bucket at level $i = 0$.

In general, if the last bucket of the last level is filled, the system is redefined, by emptying each bucket and placing the demands in the RESET bucket, with a new coloring (steps 1, 2, 4 and 8). This idea can be verified in demand #2, in which the vertices 0-1-2 and 0-1 are colored in the RESET bucket, with different green colors. In some steps of the example, as from step 2 to 3, there is a change in the number of buckets per level. When demand #3 arrives, as the RESET bucket is completely filled, the number of buckets in level $i = 0$ is recalculated (following $s = \lceil N_R^{1/d} \rceil$), taking into account the number of vertices in this RESET (named as last RESET), $N_R = 2$. Thus, there is the addition of one bucket in level $i = 0$, in the transition of steps 2 to 3. The number of buckets at level $i = 0$ is also increased from steps 4 to 5 and 8 to 9, in the example represented in Figure 1.

III. Developed RWA Planning Tool

In this section, we explain the main building blocks of our planning tool. We present the network topologies studied in this work, and characterize their physical and also some of their logical features.

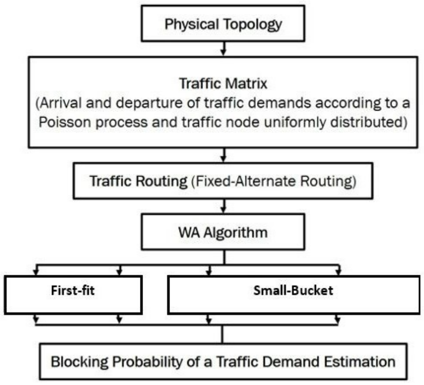

Physical Topology
Traffic Matrix
(Arrival and departure of traffic demands according to a Poisson process and traffic node uniformly distributed)
Traffic Routing (Fixed-Alternate Routing)
WA Algorithm
First-fit
Small-Bucket
Blocking Probability of a Traffic Demand Estimation

Fig. 2. Blocking probability computation.

The main building blocks of our planning tool are represented in the diagram of Figure 2:

- *Physical Topology:* in the initial phase of the simulation, the network physical topology is defined;
- *Traffic Matrix:* in dynamic networks, traffic demands arrive over time. The arrivals and departures of traffic demands, in the simulator, follow a Poisson distribution. The traffic generated by the network is uniformly distributed by each node;
- *Traffic Routing:* to calculate the optical path between a source and a destination node, the Fixed-Alternate Routing algorithm is used. This algorithm allows the calculation of the two disjoint shortest paths between any pair of nodes in the network, before the network goes into operation;
- *WA Algorithm:* after the optical path calculation for the traffic demand, a wavelength is assigned to each path. The WA algorithms that were implemented in this simulator are the First-fit and Small-Buckets algorithms;
- *Blocking Probability of a Traffic Demand Estimation:* if there is no available wavelength to allocate to the given optical path, the number of blocked traffic demands is increased. The blocking probability of a traffic demand is obtained by dividing the number of blocked traffic demands by the number of simulated traffic demands.

For the Small-Buckets algorithm, the algorithm is executed while the number of blocked demands does not reach $b_{lim} = 1000$. If the limit is reached, the blocking probability of a demand is calculated and the simulation ends. While the limit is not reached, traffic arrivals and departures are generated.

IV. Results and Discussion

In this section, the planning tool presented in section III is used to assess the performance of the Small-Buckets WA algorithm in a 16-node ring and UBN networks and also to compare it with the common First-fit WA algorithm. In particular, the blocking probability, the total number of colors, recolorings and simulation time will be assessed.

Assuming Small-Buckets as WA algorithm, the number of system levels is assumed to be $d = \lfloor \ln(N) \rfloor = \lfloor \ln(240) \rfloor = 5$ for the 16-node ring network and $d = \lfloor \ln(N) \rfloor = \lfloor \ln(552) \rfloor = 6$ for the UBN network, because this number of levels gives the minimum number of colors used [5]. In the simulations carried out, the maximum number of colors per bucket is considered so that the maximum number of colors per bucket is equal to the maximum number of wavelengths per link when the First-fit is used instead.

A. Blocking Probability and Simulation Time

In this section, the blocking probability of the networks and the simulation time of the algorithms will be analyzed. Figure 3 represents the blocking probability as a function of the average offered load per node considering $C = 48$, 56 and 64 colors and $d = 5$, for the 16-node ring network. The blocking probabilities calculated using the First-fit algorithm are also represented for comparison purposes. Some values of the simulation times are also represented.

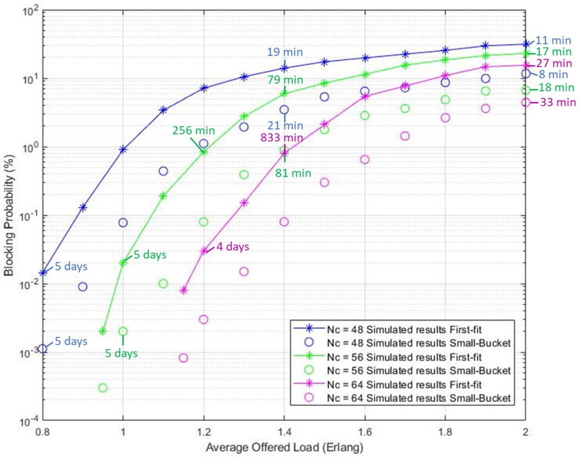


Fig. 3. Blocking probability as a function of the average offered load per node for a bidirecctional 16-node ring network with $C = 48$, 56 and 64 colors per bucket considering the Small-Buckets as WA algorithm. The First-fit is also represented for comparison purposes.

As can be seen from Figure 3, the Small-Buckets with C colors per bucket gives a lower blocking probability, compared to the First-fit algorithm with $N_c = C$. The difference between the obtained blocking probabilities, for an average load of 2 E, is approximately 8% in favour of the Small-Buckets, due to the fact that the Small-Buckets uses a much higher number of colors partitioned by s buckets to color the vertices. Since, in this scenario, the Small-Buckets algorithm uses 5 levels, the vertices can be placed in different graph coloring subspaces, thus allowing to minimize the blocking probability of a traffic demand. The comparisons between the First-fit algorithm assuming N_c wavelengths per optical link and the Small-Buckets algorithm considering C colors per bucket and a total of C_{total} colors are not completely fair, because the Small-Buckets requires a very high number of colors.

Figure 3 represents also some values of the simulation time for different average offered traffic per node and considering the Small-Buckets and First-fit algorithms. Comparing the First-fit algorithm with the Small-Buckets for the same network traffic, the simulation time obtained is very similar.

The blocking probability, as well as the simulation time, for the UBN network are also analyzed. Fig. 4 represents the blocking probability as a function of the average offered load per node considering $C = 48$, 56 and 64 colors in each bucket, and the Small-Buckets with $d = 6$ and the First-fit.

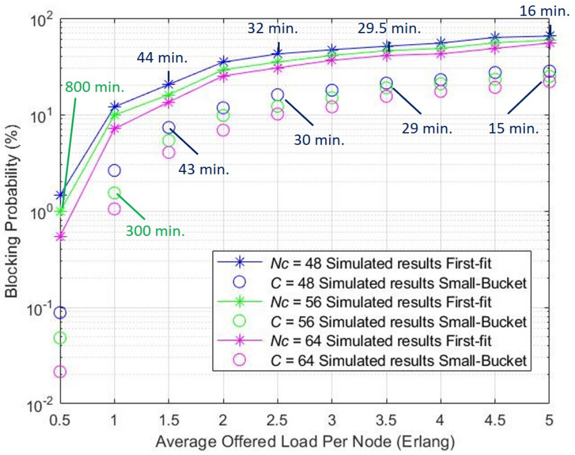


Fig. 4. Blocking probability as a function of the average offered load per node for the UBN network with $C = 48$, 56 and 64 colors per bucket considering the Small-Buckets as WA algorithm.

As can be seen from Fig. 4, once again the Small-Buckets algorithm with C colors per bucket gives a lower blocking probability, compared to the First-fit with $N_c = C$. Once again, this comparison is not completely fair, because the Small-Buckets algorithm uses a very high total number of colors, in comparison with the First-fit. Comparing the First-fit algorithm with the Small-Buckets for the same network traffic, the simulation times obtained are very similar.

B. Total Number of Colors Used and Recolorings per Update

In this subsection, the number of colors and recolorings is studied. The number of recolorings consists of the number of wavelengths reassigned in active traffic demands after the arrival of a new demand. Recolorings occur in the RESET bucket and when there are level changes. Table I represent the average number of recolorings as a function of the average offered load per node, for the 16-node ring network with $d = 5$ levels. This average number of recolorings per update are obtained after 1000 blocked traffic demands are reached.

TABLE I
AVERAGE RECOLORINGS PER UPDATE FOR $d = 5$.

A_{node}	Value	Value	Value
1.4 E	2.94	2.73	2.64
1.8 E	2.52	2.57	2.60
2 E	2.77	2.33	2.56

From [5], it can be concluded that the Small-Buckets maintains an adequate coloration of a graph by recoloring a maximum of d vertices per update, each time a new demand arrives.

In Figs. 5 and 6, the behavior of the number of recolorings as a function of the number of simulated traffic demands is studied, for $A_{node} = 0.8$ and 2 E, which correspond to $A_{network} = 96$ and 240 E, respectively.

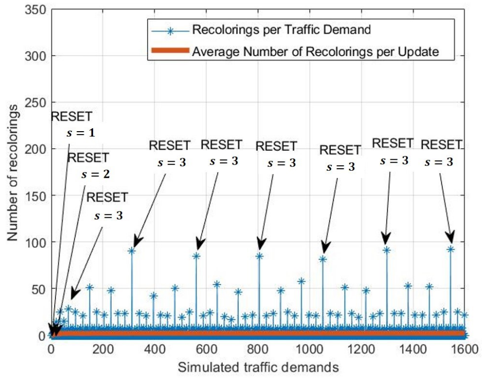


Fig. 5. Simulated number of recolorings as a function of the number of traffic demands, for $A_{node} = 0.8$ E and $C = 48$ considering a 16-node ring topology.

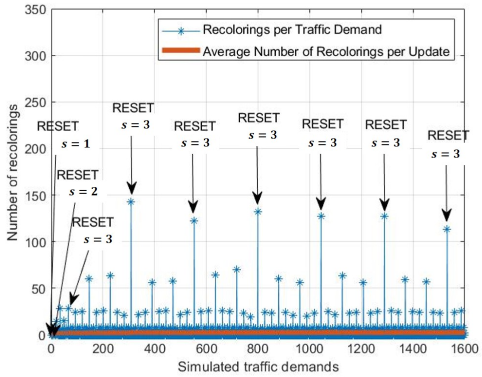


Fig. 6. Simulated number of recolorings as a function of the number of traffic demands, for $A_{node} = 2$ E and $C = 48$ considering a 16-node ring topology.

For illustrative purposes, only the first 1600 arrivals of traffic demands are shown. In these figures, the RESET situations are highlighted, as well as the evolution of the number of buckets s per level along the simulation, being this number recalculated every time a RESET occurs. From Fig. 5, it can be also observed that there are 6 big RESETS, each one corresponding to approximately 90 recolorings. A big RESET is a RESET outside the initial phase of the system. The number of recolorings has several floors, the highest one corresponds to the big RESETs, and the other ones correspond to the recolorings due to vertices level shifts. The number of recolorings due to level changes are approximately 50, 25, 10 and 3, that correspond, respectively, to changes from level 3 to 4, level 2 to 3, level 1 to 2 and level 0 to 1. In the stable phase of the system, the number of buckets per level tends to $s = 3$ and, consequently, the capacity of a bucket at level i is 3^i vertices. In Fig. 5, the average number of recolorings is also represented, and is approximately 3, after the system initial phase. This low average number of recolorings is reached due to the high number of incoming traffic demands that are placed in the available buckets without causing any recoloring.

In Fig. 6, with higher average traffic, the number of recolorings in the floors is larger than the ones found in Fig. 5. In the RESET situations outside the initial phase, there are approximately 150 recolorings, whereas in Fig. 5, this number is 90. In situations of higher traffic, the vertex coloration rate is much higher than the departure rate of traffic demands, leading to a high bucket occupancy along the simulated traffic demands evolution. With $C = 20$, and the same offered load than in Fig. 5 with $C = 48$, it can be observed that the recoloring floor levels are practically of the same magnitude. This is due to the fact that the buckets capacity, i.e. number of vertices, remains the same, despite the decrease in the number of colors per bucket.

For the UBN network simulation, it was observed that the main difference is the decrease in the number of RESETs. RESETs are less frequent because, in this scenario, $d = 6$ levels are considered compared, for example, to the 16-node ring network where $d = 5$. This additional level originates buckets with greater capacity and, thus, reduces the number of RESETs. For example, when the number of buckets per level is $s = 3$, the buckets of the last level ($i = 5$) have $C = 48$ available colors for coloring 243 possible demands ($243 = 3^5$), whereas in the 16-node ring network, the buckets on the last level ($i = 4$) have the same 48 available colors for coloring only 81 possible demands ($81 = 3^4$).

Figs. 7 and 8 shows the total number of colors used, as well as its average value, as a function of the simulated traffic demands. The RESET situations, as well as, the number of buckets per level s, are also highlighted. As can be observed in Fig. 7, the number of colors starts to increase from 0 to 70 and then at traffic demand #300, a RESET occurs and the total number of colors decreases to 48, due to the shift of all demands to the RESET bucket, which has at maximum only $C = 48$ available colors. After traffic demand #300, the number of colors used seems to have a sawtooth behaviour due to the occurrence of big RESETS, which means that the number of colors varies between 48 and 70, every 220 traffic demands. In this situation, the average number of total colors used by the Small-Buckets algorithm tends to a constant value, approximately 50 colors, while the First-fit algorithm uses only an average of 25 colors. This difference is due to the fact that the First-fit algorithm only considers a single coloring space, while the Small-Buckets considers different coloring spaces.

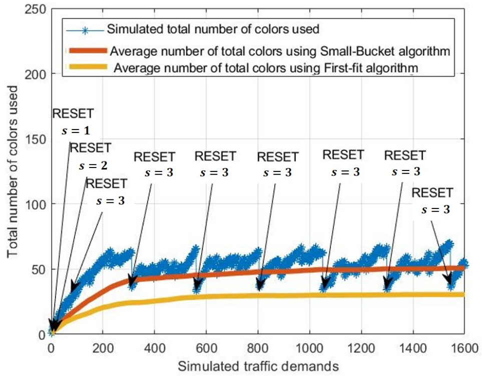


Fig. 7. Simulated number of colors used as a function of the number of traffic demands, for $A_{node} = 0.8$ E and $C = 48$ considering a 16-node ring topology.

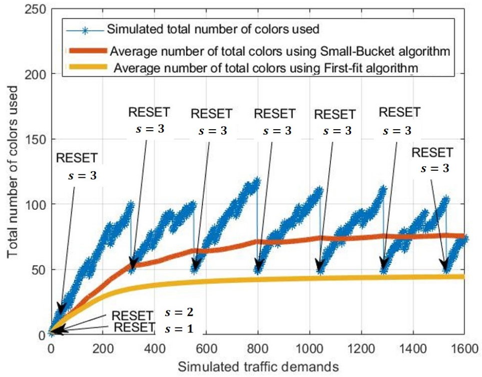


Fig. 8. Simulated number of colors used as a function of the number of traffic demands, for $A_{node} = 2$ E and $C = 48$ considering a 16-node ring topology.

In Fig. 8, the average offered load per node is greater than in Fig. 7. The main difference between these figures is the higher number of colors used in Fig. 8 along the traffic demands evolution, that leads to a higher average number of colors used. When the traffic offered increases, the number of colors is higher because the buckets are filled more quickly and, as result, more buckets and colors are used.

Comparing the performance of the Small-Buckets algorithm in the UBN network with the performance in the 16-node ring networks, it is observed that the Small-Buckets algorithm uses a lower number of colors. This is due to the large storage capacity of buckets of the last level ($i = 5$), allowing the reuse of multiple colors. For the UBN network with $d = 6$, the buckets of the last level are able to assign $C = 48$ colors to $3^5 = 243$ vertices. For example, for $A_{node} = 0.35$ E and $C = 48$, the average number of colors used is approximately 40 colors whereas, under the same conditions, for the 16-node ring networks, the average of colors used is approximately 50.

From the scenarios analyzed in this section it can be concluded that the Small-Buckets algorithm produces a large number of recolorings, that would require fast tunable transceivers, in order to tune both the transmitter and receiver for the new wavelength, which brings, besides technological issues related to the tuning speed, cost and management issues, that were not relevant when the WA is performed with, for example, the First-fit algorithm [7]. In [8], reconfiguration times of 30 μs are reported, which would probably not be enough for our scenario.

In Fig. 3, the Small-Buckets algorithm gives a blocking probability similar to the one given by the First-fit algorithm ($N_c = 48$) when $C = 40$ and $A_{node} = 1.1$ E. For this scenario, in Fig. 9, the total number of colors used as a function of the simulated traffic demands is represented.

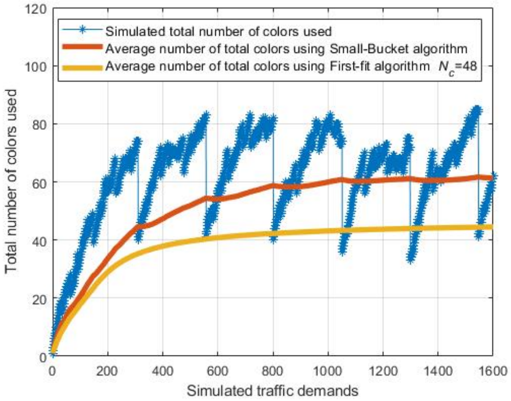


Fig. 9. Simulated number of colors used as a function of the simulated traffic demands and average offered load per node for the 16-node ring network with $d = 5$ and $A_{node} = 1.1$ E.

With this study, it is observed that to achieve the same blocking probability, the Small-Bucket algorithm uses on average a greater number of colors (approximately 12) compared to the First-fit algorithm.

V. Conclusions

In this work, the Small-Buckets algorithm was described with detail. The performance of this algorithm was assessed for the 16-node ring network through the blocking probability, number of colors used, recolorings and simulation time. A comparison with the First-fit algorithm was performed.

For the Small-Buckets, it is concluded, by considering the maximum number of colors/wavelengths per bucket (C) equal to the maximum number of wavelengths per link (N_c) in the First-fit, the blocking probability obtained using the Small-Buckets algorithm is significantly lower, because the Small-Buckets uses several buckets with capacity C, leading to a much higher average number of colors. It was also observed that the blocking probability of a traffic demand in the Small-Buckets algorithm is independent of the number of levels considered. In general, a greater number of colors is required when the Small-Buckets is considered as WA algorithm, to obtain similar blocking probabilities as the ones given by the First-fit.

The number of recolorings computed and the number of colors used are the two major weaknesses of the Small-Buckets algorithm application to optical networks, because to face recolorings with a such higher number of colors, fast tunable transceivers would be required, leading to an increase of the network cost and management. In the Small-Bucket algorithm, a demand may not be blocked because there is a possibility of recoloring. In the First-fit, there is no such possibility. The advantage of the Small-Buckets algorithm is that it allows more network dynamism due to the recoloring ability.

Acknowledgment

This work was supported under the project of Instituto de Telecomunicações UIDB/50008/2020.

References

[1] R. M. R. Lewis, *A Guide to Graph Coloring*. Switzerland: Springer, 2016.

[2] J. Simmons, *Optical Network Design and Planning*, 2nd edition, Eds. Holmdel: Springer, 2014, pp. 349-395.

[3] P. Rajalakshmi, and A. Jhunjhunwala, "Re-Routing at Critical Nodes to Enhance Performance of Wavelength Reassignment in All-Optical WDM Networks Without Wavelength Conversion," in *Journal of Lightwave Technology*, vol. 26, no. 17, pp. 3021-3029 September 2008.

[4] I. Duarte, *Graph Coloring Heuristics for Optical Networks Planning,* in *ConfTELE*, 2021.

[5] L. Barba et al., "Dynamic Graph Coloring," *Algorithmica*, vol. 81, no. 4, pp. 1319-1341, June 2018.

[6] B. Vieira, D. Conceição, J. Pedro, and J. Pires, "Evaluating the Impact of Physical Impairments on the Blocking Performance of an All-optical WDM Ring under Dynamic Traffic," *Journal of Optical Communications*, vol. 28, no. 1, pp. 66-72, Nov. 2007.

[7] N. Benzaoui, M. Szczerban Gonzalez, J. Manuel Estarán, H. Mardoyan, W. Lautenschlaeger, U. Gebhard, L. Dembeck, S. Bigo, and Y. Pointurier, "Deterministic Dynamic Networks," *Journal of Lightwave Technology*, vol. 37, no. 14, pp. 3465-3474, July 2019.

[8] L. Danh et al., "AgileDCN: An Agile Reconfigurable Optical Data Center Network Architecture," *Journal of Lightwave Technology*, vol. 38, no. 18, pp. 4922-4934, June 2020.