A STUDY ON REACTIVE AND PROACTIVE PUSH-PULL/MAKE-BEFORE-BREAK DEFRAGMENTATION FOR DYNAMIC RMSA

YAN MA

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

JULY 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Yan Ma**

Entitled: **A Study on Reactive and Proactive Push-Pull/Make-Before-Break Defragmentation for Dynamic RMSA**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________________ Chair

Dr. Jinqiu Yang ______________________________ Examiner

Dr. Dhrubajyoti Goswami______________________________ Examiner

Dr. Brigitte Jaumard ______________________________ Supervisor

Approved ______________________________

Chair of Department or Graduate Program Director

____________ 20 ______ ______________________________

Dean

Faculty of Engineering and Computer Science

Abstract

A Study on Reactive and Proactive Push-Pull/Make-Before-Break

Defragmentation for Dynamic RMSA

Yan Ma

In this thesis, we investigate several defragmentation techniques, with both proactive and reactive triggering strategies, in the context of dynamic Routing, Modulation and Spectrum Assignment (RMSA) in optical flexible networks.

Proactive defragmentation is executed periodically or according to some fragmentation degradation thresholds in order to maintain spectral defragmentation at an acceptable level, the defragmentation is independent of the request connection events. Reactive defragmentation, on the other hand, is performed when a new request is blocked due to insufficient spectral resources. In the context of dynamic traffic in a flexible optical network, we looked into different combinations of proactive/reactive push-pull and make-before-break defragmentations.

Extensive numerical results show that reactive push-pull defragmentation performs quite well in terms of network throughput and request blocking ratio. Consequently, it is efficient in order to improve network throughput. For proactive push-pull defragmentation, we investigated two different triggering events, namely, time-driven and throughput-driven. We observed that both triggering strategies have a good performance on maintaining an efficient spectrum usage in networks. Throughput-driven strategy performs better when the network is heavily loaded, whereas time-driven strategy is a better option when the network is less loaded.

Keywords: Routing, Modulation, Spectrum Assignment, Optical Network Defragmentation, Push-Pull, Make-Before-Break.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Background and Motivation

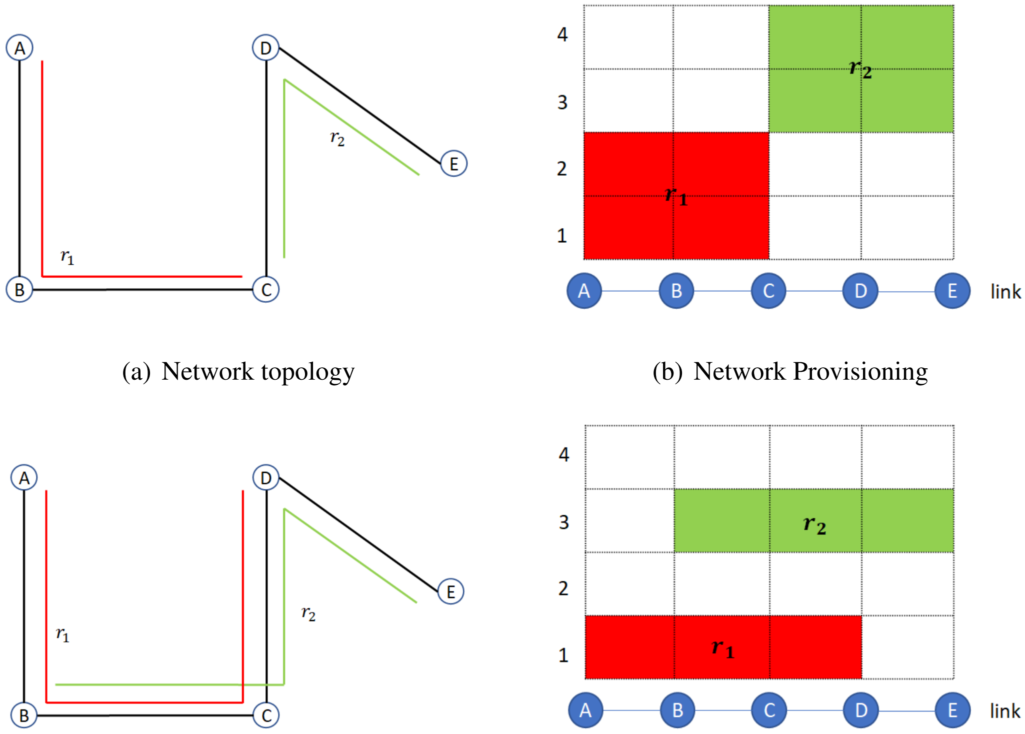
1.1.1 Introduction to Elastic Optical Networks

The rapid growth in worldwide communications and the rapid adoption of the Internet has significantly modified our way of life. This revolution has led to a vast growth in communication bandwidth in every year. Optical network technologies are important to global Internet operations since they are able to support crucial and reliable communication services [3]. In response to rapidly growing traffic demands, so far, Wavelength-Division Multiplexing (WDM) systems with more than 1,000 Gbps capacity per channel have been commercially deployed. It is expected that international bandwidth demands will be approximately 1,103.3 Tb/s in 2020 [2]. Therefore, optical networks is required to support Tb/s class transmission [11]. Nevertheless, WDM-based optical networks operate on a fixed wavelength grid, consequently, they necessarily grant traffic demands on a full wavelength even though the demands do not fill the entire capacity. This inefficient utilization of spectral resources is expected to become an even more serious issue with the deployment of higher data rates. To meet the needs of the future Internet, optical transmission and networking technologies are moving toward the goals of greater efficiency, flexibility, and scalability.

Elastic Optical Networks (EONs) are widely considered as the next generation optical networks. Different from the traditional Wavelength Division Multiplexing networks (WDM) where channels have a 50 GHz width, frequency slots have a 12.5 or 6.25 GHz width in elastic optical networks. EONs further improve the spectrum utilization with consideration of adaptive modulation. Indeed, the modulation-based EONs can reduce the allocated spectral bandwidth for shorter paths by increasing the number of modulated bits per symbol [40]. As a result, flexible network utilization efficiency is greatly improved compared to WDM based optical networks. More details on modulation are provided in Section 3.2.

1.1.2 Introduction to EON Provisioning

In EONs, routing and spectrum assignment (RSA) problem is defined as the problem of establishing connections for each request by selecting an appropriate routing path and an available spectrum allocation. The routing, modulation and spectrum assignment (RMSA) problem is an extension of RSA, with additional requirements to select an appropriate modulation format and spectrum width according to the transmission distance (see Section 3.2). Any request in EONs must use the same channel(s), i.e., the same frequency slot(s), from source to destination, this property is known as continuity constraint. Furthermore, the allocated frequency slots must be adjacent, this property is known as contiguity constraint. The concept of the contiguity and continuity constraints of the spectrum allocation is next explained with an example. Let us consider a simple network topology, shown in Figure 1, with 5 nodes and 4 links. Assume each link has a 4 frequency slot capacity.



(a) Network topology

(b) Network Provisioning

(c) Network topology

(d) Network Provisioning

Figure 1: Continuity and contiguity constraint examples

As shown in Figure 1(a) and Figure 1(b), request r_1 is following on optical routing path from node A to node C, using frequency slots 1 and 2, and request r_2 goes node C to node E using frequency slots 3 and 4. If a new request r requiring two frequency slots from node A to node E arrives, it cannot be granted even though 2 frequency slots are available on each link of the path from A to E. In other words, the new incoming request cannot use frequency slots 3 and 4 from node A to node C and frequency slots 1 and 2 from C to E. A request has to use the same frequency slots from source all the way to the destination, this is the continuity constraint.

Consider a second example in Figure 1(c) and Figure 1(d). Request r_1 from node A to D uses frequency slot 1 and request r_2 from node B to E uses frequency slot 3. If a new request r requiring two frequency slots from A to E arrives, it cannot be granted even though the same 2 frequency slots are available on each link of the path from node A to E. In other words, EONs cannot use frequency slots 2 and 4 to transmit request r because the spectrum contiguity constraint cannot be satisfied.

1.1.3 Introduction to EON Defragmentation

One of the major topics in EONs is the spectrum fragmentation. In a dynamic network scenario, where requests are allocated and disconnected in a quite random fashion, spectral resources tend to be highly fragmented because of the continuity and contiguity constraints for every request.

To address the issue of fragmentation, computer scientists and network experts are investigating dedicated mechanisms to rearrange existing traffic demands to avoid spectrum fragmentation. This is the so-called spectrum defragmentation. Several defragmentation techniques have been proposed, e.g., make-before-break, push-pull and hop-tuning. Those techniques are later detailed in Chapter 2. We give a brief description of each defragmentation technique in the following paragraphs.

Make-Before-Break defragmentation performs as follows. For each provisioned request connection, this technique finds an available alternate route, modulation and slot set (e.g., a shorter routing path). If it is better than the original one, then defragmentation can be performed in the make-before-break manner, i.e., activate a new request ('make') using the alternate route and slot set first, and keep the original request connection. Only the newly created request is properly transmitting on the new lightpath, then release ('break') the old resources.

Both push-pull and hop-tuning techniques consist of shifting the conflicting connections to free contiguous spectral resources. Conflicting connections are connections which share the expected route and slot set with the new request route and slot set. The constraint of both push-pull and hop-tuning is that the route of the processed request does not change. The difference is that push-pull only shifts conflicting requests between its adjacent request set, whereas hop-tuning can reallocate a request on any feasible spectrum location. More details on each defragmentation technique are provided in Section 3.4.1.

1.2 Dynamic RMSA Defragmentation

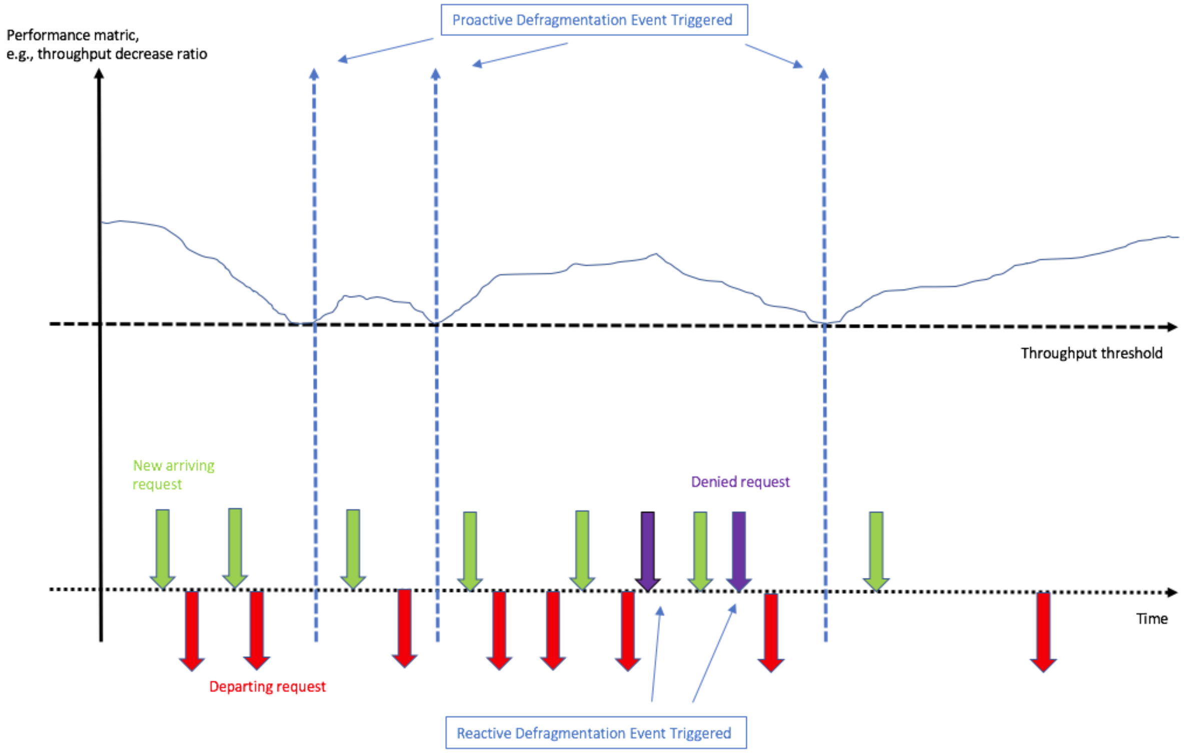

Proactive Defragmentation Event Triggered
Performance matric,
e.g., throughput decrease ratio
Throughput threshold
New arriving
request
Denied request
Time
Departing request
Reactive Defragmentation Event Triggered

Figure 2: Dynamic RMSA provisioning and defragmentation process

In this thesis, we focus on the online modulation based provisioning and defragmentation problems in EONs. Defragmentation techniques can be classified as proactive or reactive [14]. The former is executed periodically or according to some predefined thresholds in order to maintain spectral defragmentation at acceptable levels, the thresholds are independent of request connection events. The reactive methods are usually performed when a new incoming request is blocked due to insufficient spectral resources [28].

Figure 2 gives an example of RMSA reconfiguration process, X-axis represents timeline. As time goes by, there are sets of incoming requests (colored in green) and departing requests (colored in red). However, some of the new arriving requests are denied by the optical provisioning network policy (colored in purple). Y-axis measures the provisioning network performance according to some parameters, such as overall throughput, blocking ratio, time units, etc. As shown in Figure 2, when the overall throughput decreases by a given threshold, proactive defragmentation is triggered, meanwhile, the reactive defragmentation is triggered when a new incoming request is denied by the provisioning network. This thesis focuses on reactive and proactive push-pull/make-before-break defragmentation techniques.

1.3 Contributions of the Thesis

Contributions of the thesis are as follows. We developed a defragmentation framework that allows the investigation of the added value of the push-pull and make-before-break defragmentation techniques. Conclusions of the extensive computational experiments are that reactive push-pull defragmentation performs quite well in terms of improving network throughput and reducing blocked requests. For proactive push-pull defragmentation, we investigate time-driven and throughput-driven strategies, both methods have good performances with respect to network throughput. The time-driven triggering approach performs better than the throughput-driven approach in low traffic loads, but in high traffic loads, the throughput-driven approach is a better option.

1.4 Organization of the Thesis

Chapter 2 presents a literature review of related subjects, including recent studies on EON provisioning and defragmentation strategies. Chapter 3 provides a concise statement of the defragmentation problem in flexible optical networks, with the corresponding background. All the related algorithms we use for RMSA reactive defragmentation are presented in Chapter 4, complexity analysis of algorithms are also discussed in this chapter. Chapter 5 introduces an RMSA proactive defragmentation model based on make-before-break with and without push-pull. Finally, Chapter 6 conducts a numerical analysis of the performance of the designed algorithms in the previous chapters, as well as the characteristics of the solutions. Conclusions and future work are discussed in the last chapter.

Chapter 2

Literature Review

In this chapter, we present related works on EON provisioning and defragmentation. We first review the studies on provisioning strategies in EONs, and then the recent works in flexible optical network key defragmentation techniques, i.e., make-before-break, push-pull, and hop-tuning.

2.1 EON Provisioning Strategies

The R(M)SA problem can be classified under one of the two board versions: *offline* R(M)SA (or *static* R(M)SA), whereby the traffic demands are known in advance and *online* R(M)SA (or *dynamic* R(M)SA), in which a sequence of client requests arrive in some random fashion [29]. The next two sections discuss spectrum management techniques for *online* and *offline* R(M)SA respectively.

2.1.1 Offline R(M)SA

Offline R(M)SA with ILP (Integer Linear Programing)

The static RSA problem has been formulated as an ILP that returns the optimum solution through a joint routing and spectrum allocation [4] and [15]. The objective of the ILP is to minimize the utilized spectrum, with the constraints of spectrum continuity and contiguity constraints. In order to reduce the complexity of the combined RSA, Christodoulopoulos *et al.* [4] present a decomposition ILP model which breaks RSA into its two substituent subproblems, namely, available routing path searching and spectrum allocation, and solves them sequentially. To feed the sequential algorithm, two ordering policies are proposed, i.e., most request demand first and shortest path first. Results indicate that the proposed sequential heuristic combined with an appropriate ordering can give close to optimal solutions in low running times.

An offline version of the RMSA problem was studied in [5]. In this problem, request demands are mapped to a modulation level based on the requested data rate and the distance of the path over which it is routed, with the mapping function provided as input to the problem. In [5], the path for each request is predefined, and then the problem was decomposed into two subproblems, routing and modulation level (RM) and spectrum assignment (SA) and solved sequentially (RM+SA) using ILPs.

The above ILP formulations are able to find optimum or near-optimum solutions for small networks. However, they are not scalable to large networks, e.g., in a simulation of DT (Deutsche Telekom) network topology with 14 nodes and 46 links, the combined RSA ILP model in [4] and [15] could not return a solution in a reasonable time, while the sequential RSA ILP model in [4] took several hours [40].

Heuristic Algorithms for Offline R(M)SA

To solve the R(M)SA problem efficiently, several heuristic algorithms have been proposed to serve each connection request sequentially in offline R(M)SA scenarios.

Wang *et al.* [35] developed two heuristic algorithms to solve offline RSA efficiently. The first algorithm is referred to as shortest path with maximum spectrum reuse (SPSR). The algorithm first sorts the requests in decreasing order of their demands, then uses the shortest path routing and first-fit spectrum allocation strategy to assign frequency slots to demands. The second algorithm, called balanced load spectrum allocation (BLSA), considers the k shortest path set as a candidate for each request and selects the link set with the minimum spectrum usage, so as to balance the use of the spectrum across the network links.

A greedy algorithm with consideration of modulation is proposed in [4] in order to address offline RMSA problem. The algorithm firstly sorts the requests in decreasing order based on their demands or the length of their shortest paths, then solves the routing path selection problem and spectrum allocation problem sequentially. Another heuristic algorithm called adaptive frequency assignment with collision avoidance (AFA-CA) is proposed in Klinkowski *et al.* [16], the algorithm processes the requests in an order which is based on link traffic metric to avoid selecting paths that will result in congested links.

2.1.2 Dynamic R(M)SA

Because of the real-time nature of the problem, RSA algorithms in a dynamic traffic environment must be simple and fast [27]. Since combined routing and spectrum assignment is a hard problem, most studies in dynamic RSA planning decompose RSA into sequential routing and spectrum assignment problems and solve them separately [27]. Moreover, most of the studies on online modulation based spectrum allocation have introduced heuristic algorithms [2].

In dynamic R(M)SA scenarios, the candidate routing paths can be predefined and ordered. Therefore, the spectrum allocation policy, which determines which set of available (satisfying spectrum continuity and contiguity constraints) slots are assigned to a request, is crucial to the performance of an online R(M)SA algorithm. In spectrum allocation algorithms in the context of R(M)SA, a first-fit policy (in [29] and [34]) selects the lowest slot index set. A random-fit policy (in [39]) randomly allocates one of the available allocations, whereas a best-fit policy selects the indexed slot set with the smallest size.

An improvement in the operation of the first-fit policy has been proposed in Almeida *et al.* [1]. The authors proposed an evolutionary algorithm to search for the most feasible spectrum ordering for first-fit so as to minimize the blocking probability, the study showed that the algorithm has a significant reduction on the blocking probability compared to the conventional first-fit policy.

The study in [34] investigated the optimal slot width for first-fit policy under the dynamic traffic under a hypothesis that each request is routed on its shortest path, and the first-fit policy was used for spectrum allocation selection. The author finds that the best performance, in terms of the blocking probability, is achieved when the slot width is equal to the greatest common factor of all the request frequency slot demands.

On the other hand, Wan *et al.* [30] proposed several first-fit based algorithms in dynamic RSA. A request routing path is determined by different algorithms, including *k* shortest path (K-SP), Modified Dijkstra Shortest Path (MDSF), and Spectrum-Constraint Path Vector Searching (PVC). The first-fit policy is used to assign frequency slots to the request. In addition, a routing path selection based on dynamic ant colony optimization (ACO) algorithm is proposed in [36], and the authors find that the proposed algorithm performs better than (K-SP) in terms of request blocking ratio.

We discussed in the above paragraphs for the basic online RSA problem. For RMSA problem, the main difference is that the modulation should be defined for each request in order to determine the slot demands based on their data rate and routing path. Therefore, the spectrum allocation policy must search for different modulation formats which are dependent on the length of the path considered.

In [26], an algorithm is proposed to address online RMSA problem. The algorithm precomputes the paths for each source-destination pair, and order them in decreasing length. In order to locate the request on the spectrum allocation, it employs first-fit policy and sequentially considers each path until a feasible (i.e., which satisfies the spectrum constraints) frequency slot set that is able to accommodate the request data rate over the selected path length. A link load balance RMSA online algorithm is studied in [37], the heuristic algorithm grants requests with the constraint to balance the link loads and it works as follows: for each modulation format, each link in the network is assigned a weight. The weight is equal to the ratio of the required slots over the number of free slots on the link. Next, for each modulation, a modified Dijkstra's algorithm is used to find the minimum cost path with the feasible contiguous spectrum for the request. Finally, the path, and modulation with the smallest cost (if any is found) are assigned to the request.

2.2 EON Defragmentation Strategies

EONs grant requests on contiguous frequency slots and feasible routing paths. Therefore, dynamically setting up and tearing down requests can generate the bandwidth fragmentation problem [3]. It is the condition where available slots become isolated from each other by being misaligned along the routing path or discontiguous in the spectrum domain. Thus, it is difficult to utilize them for upcoming connection requests. Therefore, spectrum defragmentation strategies are necessary to be investigated. Defragmentation usually involves rerouting or spectral reallocation of existing requests and may require a large amount of time to converge [28]. During the defragmentation process, the invoked requests would be affected and the *Quality of Service* would be deteriorated. This is referred to as traffic disruptions. Hence, one of the key operational requirements is to not disrupt the service during the defragmentation phase, or at least minimize disruptions. Based on traffic disruption, defragmentation policies are categorized into the non-hitless (the defragmentation with traffic disruptions) and hitless defragmentation (the defragmentation without traffic disruptions) strategies, which are discussed in the following.

2.2.1 Non-hitless Defragmentation

Re-planning technique is proposed in [22]. The authors, in particular, formulate the network defragmentation problem in EONs, model it, and propose two heuristic algorithms, namely, greedy based (Greedy-Defragmentation) and shortest path based (SP-Defragmentation) heuristic algorithms. The authors compare these algorithms with an ILP model, find that greedy based algorithm is closer to the optimal solution but has higher defragmentation traffic disruptions than the shortest path based algorithm. In [9], the authors examine defragmentation in practice by re-planning requests while also considering the advantages obtained by different channel spacing selections. Eira *et al.* [9] finds that the gain of re-planning defragmentaion strongly depends on the spectral widths.

Spectrum partition technique is introduced in [31], this technique slices spectrum into several parts and allocates requests in one of the parts according to some predefined rules. In [31], Wang and Mukherjee proposed several spectrum partition methods, namely, i) complete sharing with partitioning (CS), where all the granted requests are re-allocated to one part of the network (high and low spectrum locations); ii) pseudo-partitioning (PP), where the requests with high and low demands are allocated at the high and low spectrum location respectively; and iii) dedicated partitioning (DP), where each partition of spectrum carries a uniform data rate and where they seek an optimal partitioning. The simulation shows that DP is a prior option since it is fair for requests with high and low demands, and the request provisioning efficiency is significantly better than other proposed methods (CS and PP), especially when the network becomes loaded.

In [41] and [43], the authors presented a study on the proactive defragmentation policies. They considered the questions of when, what, and how to defragment. Firstly, they choose a portion of existing requests as candidate rerouting request sets, using connection selection strategies and then determine how to reroute them with the defragmentation based RSA. Finally, in order to minimize the number of request migrations and minimize traffic disruptions, dependency graphs are introduced. The dependency graph is a directed graph which finds the precedence relationships between requests and is used to minimize defragmentation disruptions. For example, if there is an arc from request A to B, it means that request B has to be processed before A in the defragmentation. In addition, the authors tested the proposed algorithms under different network initial status in [43].

In [18], the authors compared the performance of the non-hitless defragmentation (including rerouting and non-rerouting policies) in EONs, with both reactive and proactive approaches, in terms of request blocking ratio, and bandwidth fragmentation ratio, etc. Simulation results show that proactive defragmentation has a better performance in low traffic loads, but in high traffic loads, the reactive approaches are better options.

2.2.2 Hitless Defragmentation

Hitless defragmentation is a defragmentation strategy that works continuously in EONs without traffic interruptions. It advocates retuning the granted requests to fill the gap the provisioning network, in order to make a compact spectrum utilization. The following discussion is about two retuning approaches, namely, push-pull and hop-tuning.

The push-pull technique was proposed in [7], in which the authors considered push-pull as a cheap and non-disruptive defragmentation technique because it does not require additional transponders and does not disrupt the other request connections. In order to evaluate the performance of push-pull, a linear programming model is proposed, and the simulation is based on Telecom Italia Sparkle topology. Results show that spectrum defragmentation operation is achieved without additional cost transponder support and it successfully avoids connection disruptions.

In an in-depth study, Cugini *et al.* [8] focused on the feasibility demonstration and performance evaluation of push-pull technique on different transmission and detection strategies. Technological and impairment-related issues are also taken into account. The results show that actual request frequency re-tuning required just few tens of milliseconds. Therefore, the authors believe that push-pull technique can be considered for high frequent utilization.

In [33], Wang and Mukherjee proposed a heuristic algorithm based on push-pull. The authors compared its defragmentation performance with first-fit and spectrum partition strategies. Simulation results showed that the algorithm performs better in terms of the blocking ratio. Specifically, they find that a novel framework of provisioning using reactive defragmentation perform quite good when the network has a light load.

Coudert *et al.* [6] improved push-pull algorithms in [33] in terms of minimizing delay. Furthermore, a heuristic push-pull algorithm based on the shortest available path is proposed. The authors simulated the algorithms on different undirected networks and tested the push-pull algorithms under Spectrum Blocking Ratio (SBR), Average Delay (AD), and Average Shifted Distance (ASD). Results show that the push-pull algorithm based on the shortest available path always has the best SBR but the worst AD and ASD. However, the push-pull algorithm based on minimum delay has the best ASD but not necessarily best AD and SBR.

Push-Pull is a cheap, fast and non-disrupted defragmentation technique which means that it can complete the defragmentation process in a very short time without affecting the granted requests and needing costly additional transponders. However, it has limitations of not being able to sweep over other granted requests and not allowing rerouting.

In order to overcome the above limitations, Proietti *et al.* [24] proposed a new hitless defragmentation technique, where an optical request can be allocated to any desired spectral location with very short latency times (less than $1\mu s$) based on using very fast tuning transponders. This technique is named as hop-tunning.

Zhang *et al.* [42] proposed two hitless defragmentation algorithms, named as maximum spectrum rejoin (MSR) and minimum number of operations (MNO), in order to maximize spectrum rejoins and to reduce the number of operations. The MSR algorithm is applied in both hop-retuning and push-pull techniques, while the MNO is only applied for hop-tuning. Their results indicate that both algorithms reduce spectrum fragmentation in EONs.

Moniz *et al.* [19] made a comparison between push-pull and hop-tuning techniques under a unified framework. The authors proposed integer linear programming models and heuristic algorithms to study the effectiveness of these techniques and presented a performance analysis based on spectrum usage. The relative performance of the different defragmentation techniques was validated on different network topologies. This paper also validated that the spectrum gains associated with them are greatly influenced by the initial loaded network.

However, as stated in [24], hop-retuning technology is hard to deploy in EONs. This is due to the sensitivity of the spectrum wavelength modulation. Therefore, hop-retuning technology is not preferred as a defragmentation approach for a fine granular grid.

In online RMSA environment, we only consider hitless defragmentation techniques. As the hop-tuning is costly to deploy and has limitations on defragmentation performance (only reallocation, no rerouting of a request) [24]. Therefore, hop-tuning can be considered as a restricted make-before-break technique with no change in the original routes of requests [40]. In addition, according to the conclusion from Proietti *et al.* [24], hop-tuning is sensitive to request modulation. Based on the mentioned constraints of using hop-tuning, in this thesis, we only consider push-pull and make-before-break defragmentation techniques in RMSA.

Chapter 3

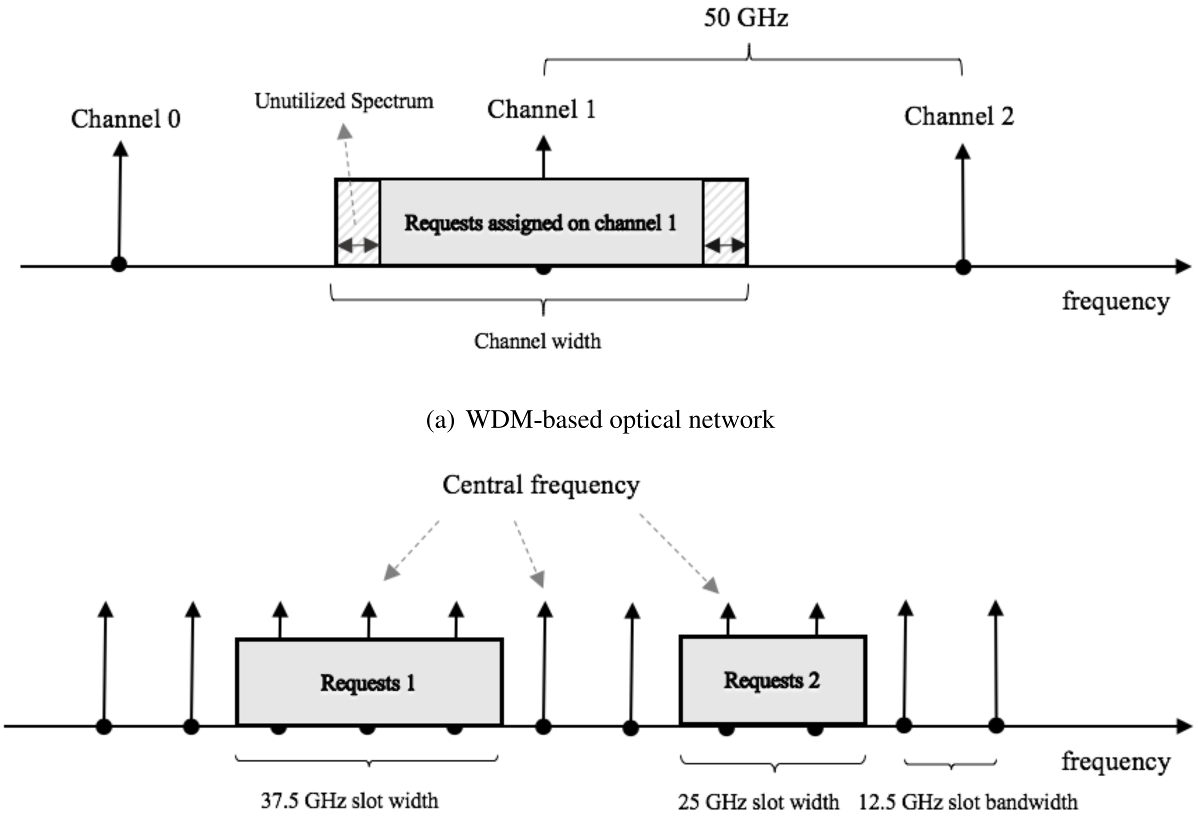
Defragmentation in Elastic Optical Networks

This chapter is devoted to the detailed description of the defragmentation problem in flexible optical networks. We gradually introduce the various technical components of the problem and describe all the notations that will be used in the sequel of the thesis. Finally, we formally state the sub-problems we will study in order to be able to address the defragmentation problem.

3.1 Elastic Optical Networks

The traditional WDM-based optical network divides the spectrum into non-overlapping channels. Each channel has its central frequency. The spacing between two adjacent central frequencies is 50 GHz, which is specified by International Telecommunication Union (ITU)-T standards [40]. As shown in Figure 3(a), if a request connection only requires a fraction of the available bandwidth of a channel, that channel would not be used efficiently since there would be a big wasted spectrum in it and no other requests would be able to use it for their transmission.

An elastic optical network has the capability to slice the spectrum into slots with finer granularity than WDM-based networks. A frequency slot is defined by its nominal central frequency in the whole spectrum range and its slot width [28]. As shown in Figure 3(b), the width of a frequency slot depends on the transmission system. There are two standard values today: 12.5 Ghz and 6.25Ghz. In the example of Figure 3(b) [28], one frequency slot is 12.5 GHz. According to the bandwidth demand of a connection request, a group of frequency slots needs to be assigned consecutively in the frequency domain. Since the spectrum assignment is more flexible compared to traditional WDM-based optical network, EONs improve the spectrum utilization significantly.



(a) WDM-based optical network

(b) Elastic optical network

Figure 3: WDM-based and elastic optical network [28]

3.2 Elastic Optical Network Modulation

The traditional WDM-based optical network grants requests on routing paths without considering the appropriate modulation technique, which leads to an inefficient utilization of the spectrum resources [40]. The elastic optical network grants requests while taking adaptive modulation into consideration to further improve the spectrum efficiency. The modulation-based spectrum allocation scheme improves the spectrum efficiency, as an advanced (higher level) modulation can reduce the transmitted symbol rate and achieve higher spectrum efficiency [40], which reduces the request slot requirements.

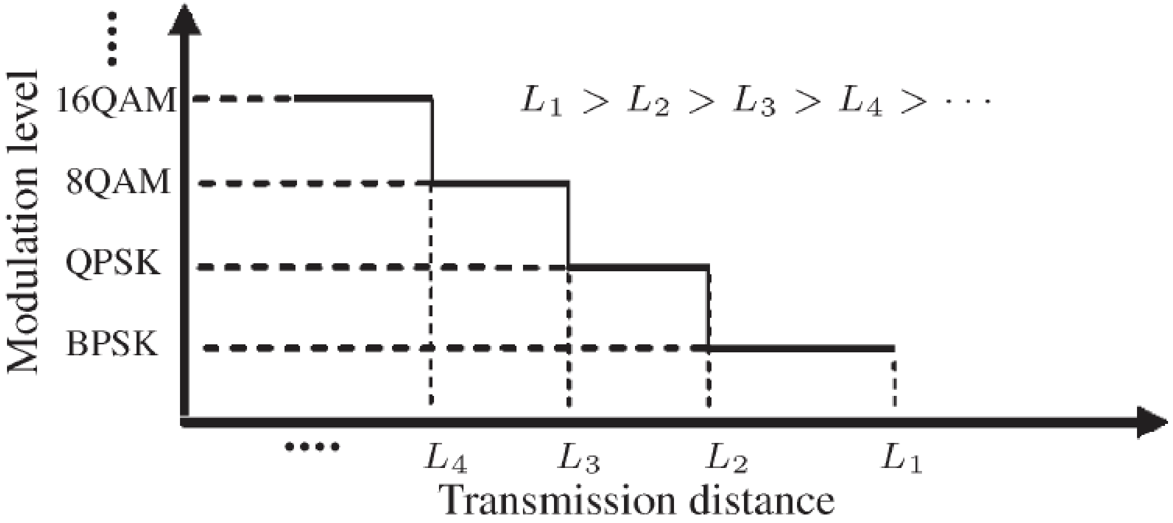


Figure 4: Modulation level versus transmission distance [3]

Jinno *et al.* [15] have presented a distance adaptive spectrum allocation scheme that adopts a high-level modulation format for long distance paths, and a low-level modulation format to shorter paths. As the optical signal-to-noise ratio (OSNR) tolerance of 64-QAM is lower than that of QPSK [40], it suits shorter distance requests as shown in Figure 4. In summary, a high-level modulation format with narrow spectrum and low OSNR tolerance maybe selected for a short path, whereas a low level modulation with a wider spectrum and high OSNR tolerance may be used for a longer path [27].

3.3 Elastic Optical Network Provisioning

Elastic Optical Network provisioning problems can first be partitioned into a routing path and modulation selection sub-problem, then into a spectrum assignment sub-problem and solved sequentially [40]. In the routing path and modulation selection sub-problem, as we discussed in Chapter 2, the candidate routing paths can be predefined. k shortest path (KSP)-based routing path selection is discussed in [4] and [29], whereas KSP with load balancing constraint is discussed in [15], in which it determines the routing by balancing the load within the network, in order to potentially minimize the spectrum usage in the network. It was shown that KSP-based routing path selection has a better performance in terms of total spectrum usage, whereas load-balanced routing path selection performs better in terms of minimizing used spectrum index in the network.

After the routing path is selected, the spectrum allocation problem has to be solved by using some spectrum allocation selection policies, e.g., first-fit, random-fit, etc. The first-fit spectrum allocation policy always attempts to choose the first feasible indexed slot for a request connection and allocates it. By selecting spectrum allocations in this way, provisioned requests are compacted into a relatively smaller number of spectrum slots, leaving a larger number of spectrum slots available for future use. This policy is widely used in online R(M)SA provisioning due to its lower call blocking probability and computation complexity.

In this thesis, we use k shortest path-based first-fit algorithm for online RMSA planning, since this is a cheap and easy online RMSA planning strategy. The idea is as follows: Considering a new incoming request r with its source SRC_r, destination DST_r and data rates. Note that the spectrum requirements, i.e., number of frequency slots d_r^m, depends on the assigned modulation m. We can easily choose one path from the pre-calculated k-shortest path set. We assume k to be a small integer, and the k-shortest paths using Yen's algorithm [38]. For each candidate routing path, we assign the highest-level modulation to r in order to get less slot demands. Then, we choose the 1st feasible spectrum allocation which satisfies the contiguity and continuity constraints, otherwise, the request is denied. The pseudo-code is presented in Algorithm 1 and denoted by KSP-FF.

Algorithm 1: KSP-FF(r), online RMSA provisioning

```
Input  : A new incoming request $r$ with source $\mathrm{SRC}_r$, destination $\mathrm{DST}_r$ and its data rate,
         an optical provisioning network
Output: $r$'s provisioning in the network if it is provisioned.
1  KSP = find $k$ shortest paths from $\mathrm{SRC}_r$ to $\mathrm{DST}_r$;
2  $R^G$ = provisioned request connection set;
3  for ( $p \in KSP$ ) {
4      $d_{temp}$ = slot demands based on a modulation $m$ ($m$ is the highest-level modulation
         associated with routing path $p$'s distance);
5      for ( every spectrum allocation $s \in S$ ) {
6          if request $r$ can be provisioned on spectrum interval $[s, s+d_{temp}-1]$ with routing
             path $p$ then
7              set $r$'s routing path as $p$;
8              set $r$'s spectrum allocation as $s$;
9              set $r$'s modulation as $m$;
10             set $r$'s slot demands as $d_{temp}$;
11             $R^G = R^G \cup \{r\}$;
```

Here is the running time complexity analysis for Algorithm 1:

Step 1 finds the k shortest path (KSP) for the new incoming requests from source SRC_r to destination DST_r using Yen's algorithm [38], which has a running time $O(kN(L+N\log N))$, in which L represents the number of links and N is the number of nodes, k is an integer. **Step 3** iterate all the candidate routing paths $p \in$ KSP. **Step 4** assigns slot demands to r based on modulation m, where m is the highest-level of modulation can be used on p. **Step 5** iterates spectrum allocations $s \in S \setminus \{|S|-d_{temp}+1, \ldots, |S|\}$, where $|S|$ represents the link capacity, i.e., the number of frequency slots. **Step 6** checks if the spectrum allocation s is feasible on the selected routing paths. The time complexity depends on r's path size and the slot requirements. **Step 7-12** set the routing path, spectrum allocation and modulation format for r and return.

The overall running time complexity is $O(k(NL+N^2\log N+|S|^2L))$. In this thesis, we consider a capacity of 400 slots on each link ℓ ($\ell \in L$), therefore, we consider link capacity S (400 slots) as a constant. Therefore, we can simplify the time complexity as $O(k(NL+N^2\log N))$.

3.4 Elastic Optical Network Defragmentation

3.4.1 Generalities

Any request connection in EONs must satisfy the spectrum contiguity and continuity constraints. Therefore, dynamically setting up and tearing down request connections with different demands will unavoidably generate a network fragmentation problem [40]. As a result, slots are isolated from each other by being misaligned along the routing path or discontiguous in the spectrum domain. Thus, it is difficult to utilize them for upcoming connection requests because of the spectrum constraints. A request is blocked if there is no available slot set can fulfill its required demands. This is referred to as network fragmentation.

Therefore, dedicated spectrum defragmentation mechanisms have begun to be investigated. Their aim is to rearrange existing request connections to make room for the other potential forthcoming request connection(s). As rearrangement usually involves rerouting or spectral reallocation of existing connections, defragmentation techniques may require a large amount of time to converge [27]. Hence, one of the key operational requirements is to not disrupt the service during the reconfiguration phase (or at least minimize its effects) [40].

An example is shown in Figure 5. The X-axis represents links in network topology and the Y-axis represents frequency slots. Figure 5(a) is the optical provisioning network before defragmentation, Figure 5(b) shows the network after defragmentation. As we can observe from Figure 5(a), the network can hardly grant a new incoming request even though it has a lot of vacancy spaces. Then, we follow a predefined rule in order to reconfigure the provisioned requests on better spectrum locations. Finally, a big set of contiguous slot set is made after the defragmentation, as shown in Figure 5(b), so that the optical provisioning network can easily grant a new request which satisfies the spectrum constraints. This process is the EON defragmentation.

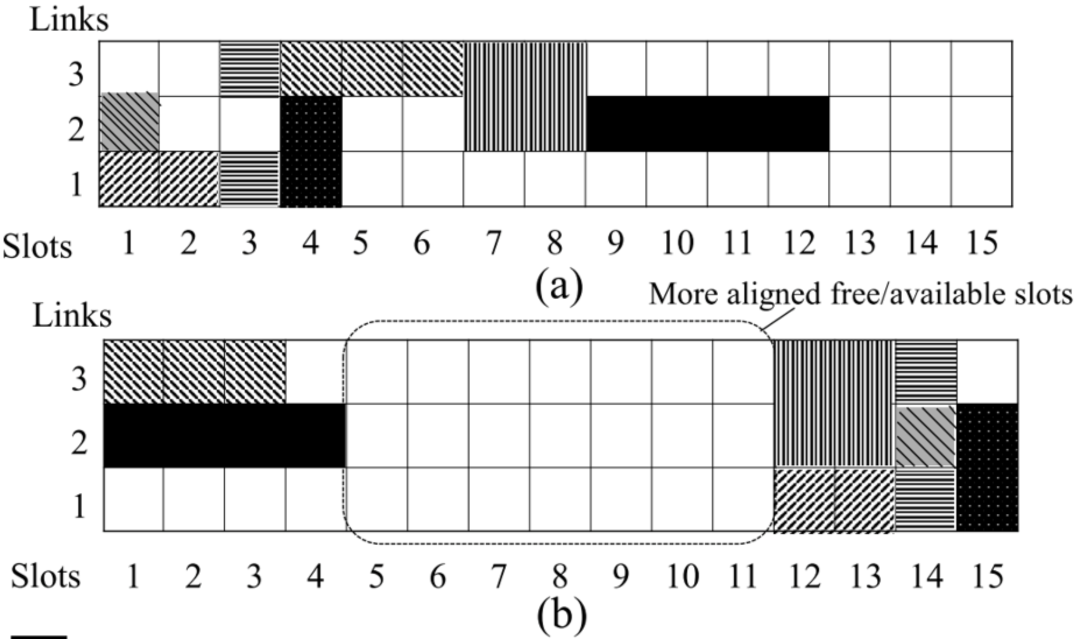

Links
3
2
1
Slots 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(a)
More aligned free/available slots
Links
3
2
1
Slots 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
(b)

Figure 5: Elastic optical network fragmentation and defragmentation example [10]

Many R(M)SA defragmentation policies have been proposed to address EON fragmentation. Here are examples to explain how these methods work. Figure 6 shows several spectrum defragmentation policies, namely, re-planning, make-before-break, push-pull and hop-tuning.

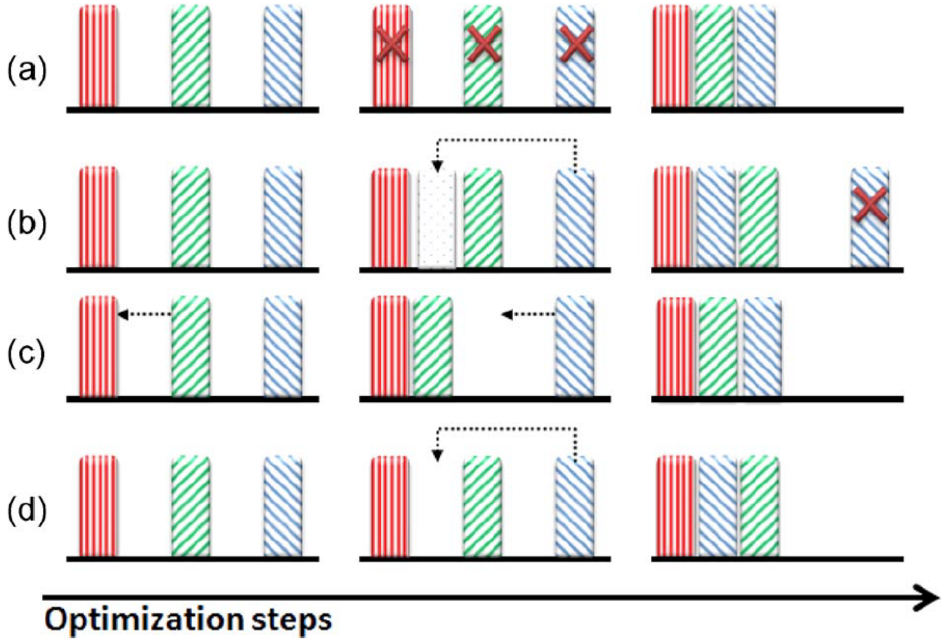


Figure 6: Elastic optical network defragmentation policies [28]

Re-planning technique was discussed in [22]. This scheme achieves defragmentation by moving a request to any desired spectrum location on any feasible routing path. An application of the re-planning solution is exemplified in Figure 6(a). Therein, all the requests are reconfigured by using the re-planning defragmentation policy. The re-planning offers the most flexibility in reassigning connections (in terms of available paths and spectrum resources) and is easy to deploy. However, this method introduces request disruptions and takes a long time to complete the defragmentation process [3].

The make-before-break approach presented in [25]. Figure 6(b) depicts the defragmentation processes of make-before-break. This scheme carries out the defragmentation operation by moving a request to any desired spectrum location on any feasible routing path. However, the desired request connection should be established properly before make-before-break releases the original request connection. As shown in Figure 6(b), the desired location of the request in blue is between the request in red and the request in green. The make-before-break policy establishes the blue request on its desired position first and then releases the original request connection.

The push-pull approach is proposed in [8]. This technique achieves defragmentation by shifting the invoked request to nonconflicting and contiguous slots without changing the request original routing path. Figure 6(c) illustrates the equivalent steps of push-pull defragmentation. As shown in Figure 6(c), in order to make room at a higher indexed spectrum allocation, the request in green is shifted to lower the spectrum allocation as much as possible and is allocated just adjacent to the request in red, and then the blue request follows the same policy and is allocated next to the green request.

The hop-tuning technique is introduced in [24]. This scheme carries out the defragmentation operation by moving the connection to any desired spectrum location subject only to the constraint of not changing its initial path. Figure 6(d) depicts the steps for hop-tuning. As seen in Figure 6(d), the request in blue can be reconfigured directly to the spectral allocation between the request in red and the request in green in a very short time and without affecting other provisioned requests [28]. Although the defragmentation process of hop-tuning is very fast ($< 1\mu s$) and is an efficient defragmentation technique [28], it requires high-level hardware support and it is sensitive for a fine granular grid [24].

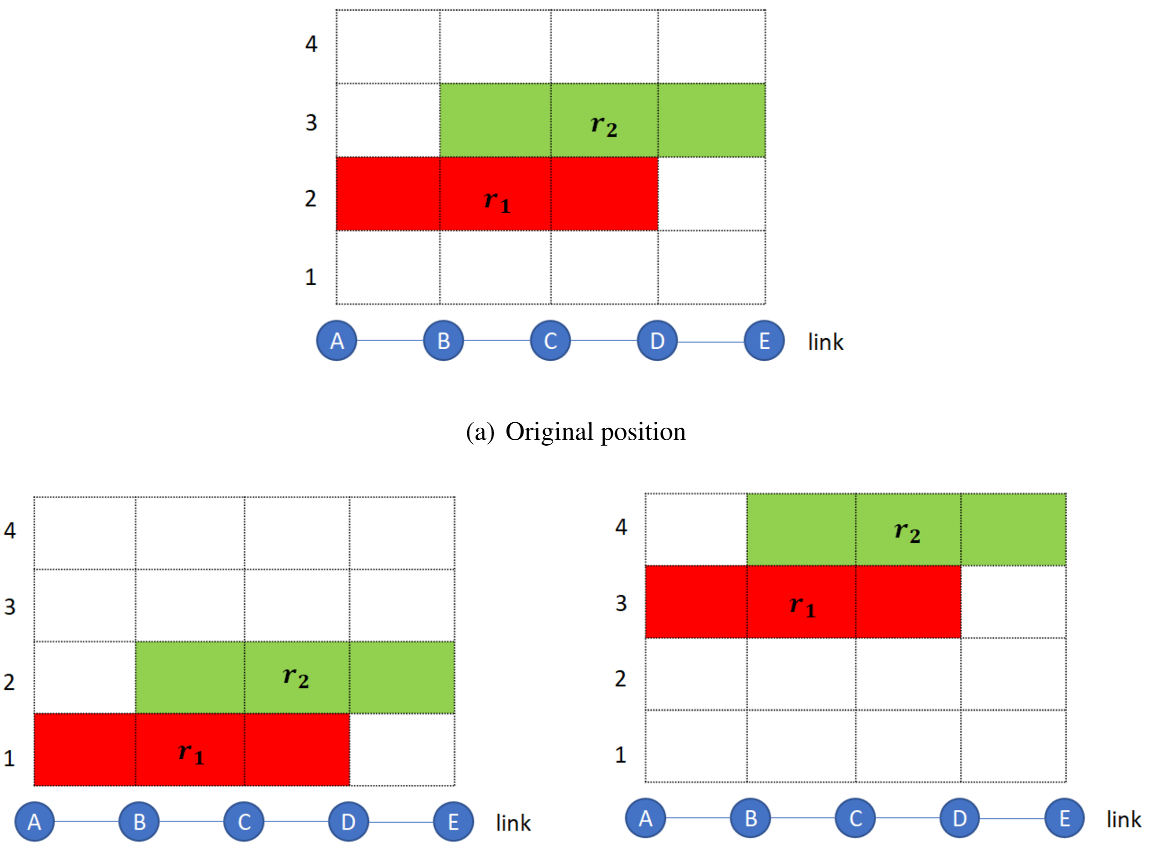
3.4.2 Push-Pull Defragmentation

In this section, we will introduce all the definitions and notations throughout the thesis.

We consider an elastic optical network represented by a directed multi-graph $G = (N, L)$ where N is the set of nodes (indexed by n) and L denotes the set of links (indexed by ℓ). Different links may exist between two nodes in order to model different fiber links.

Let S_ℓ denote the transport capacity of link ℓ, measured by the number of frequency slots in the context of RMSA. Let R be the set of provisioned requests (indexed by r). Request r is characterized by its source (SRC_r), destination (DST_r), and data rate. A provisioned request has its modulation format m, a routing path p_r (indicates a set of links that the request is using) and slot demand d_r^m (the number of slots is being used by r). We refer to $b(r)$ and $e(r)$ as the beginning slot index and ending slot index for a provisioned request r respectively, and $e(r) = b(r) + d_r - 1$.

We borrow some definitions and notations from [33]. For a request r with its routing path, the conflict set is defined as the set of provisioned requests that use routing paths sharing at least one link with r, denoted by $CS(r)$. If the request is provisioned, the new request r will partition the set $CS(r)$ into two subsets, requests above r and requests below r, i.e., every position in the spectrum of r corresponds to a partition $A \cup A'$ of $CS(r)$. With respect to the possible shiftings of the requests, we have the Δ and ∇ states. Δ state is the state of the network after shifting all the requests down (towards slot index 1) until they are blocked. ∇ state is the state of the network after shifting all the requests up (towards the link capacity S_ℓ) until they are blocked. A partition $A \cup A'$ gives the largest free interval when we shift the requests that are above (i.e. A) to the Δ state and those below (i.e., A') to the ∇ state. In each partition $A \cup A'$ we call the floors of a position α, where $\alpha = (A, A')$, which is defined as $f(\alpha) = \max\{e(r) : r \in A\}$. The floor star of a position $\alpha = (A, A')$ is $f^\star(\alpha) = \max\{e_\Delta(r) : r \in A\}$. On the other hand, we define the ceiling of a position α as $c(\alpha) = \min\{b(r) : r \in A'\}$. The ceiling star of a position α is $c(\alpha) = \min\{b_\nabla(r) : r \in A'\}$. We define $b_\Delta(r)/e_\Delta(r)$ and $b_\nabla(r)/e_\nabla(r)$ which are the beginning/ending frequency slots of request r in the Δ state and the ∇ state respectively. An illustrative example of the above notations is presented in Figure 7:



(a) Original position

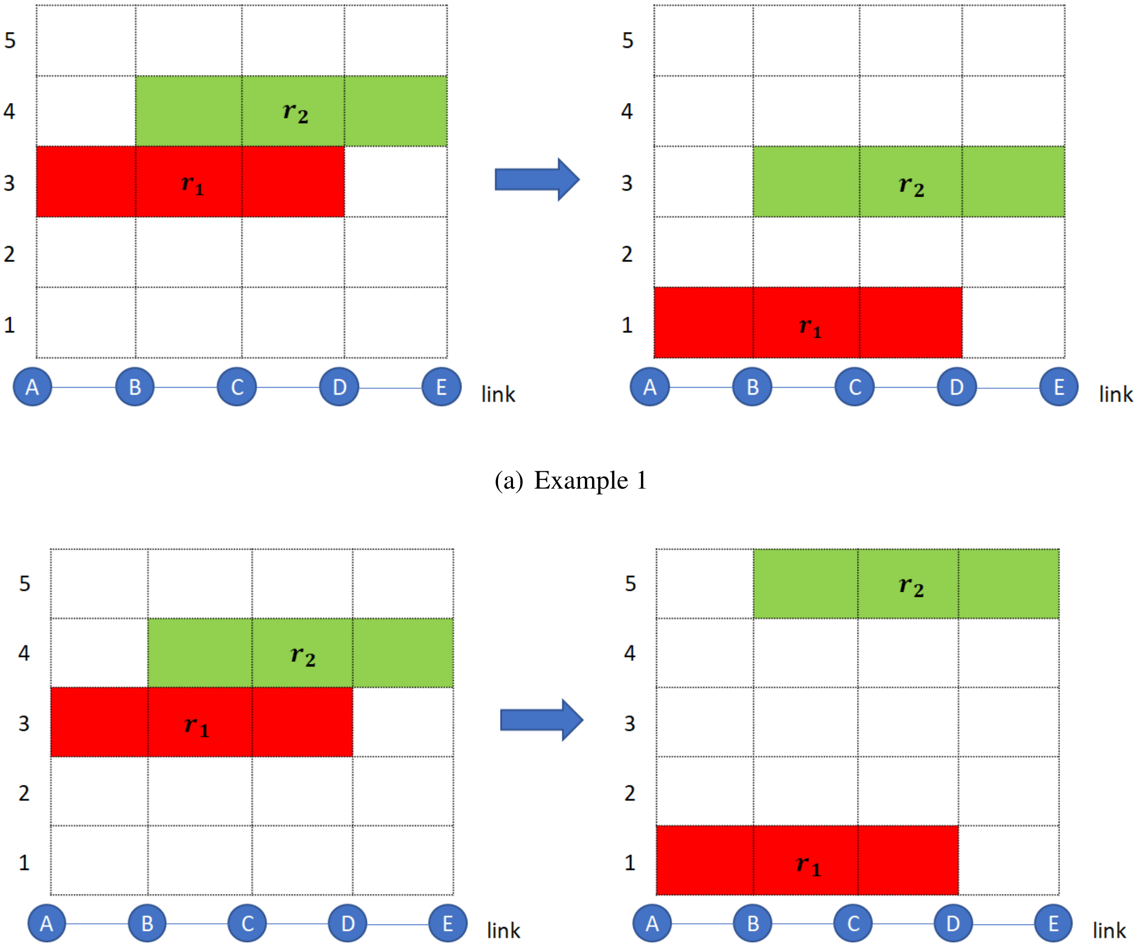
(b) Δ position

(c) ∇ position

Figure 7: Shifted position with push-pull [6]

As shown in Figure 7(a), the original optical provisioning network has 5 nodes and 4 links, and all the links have the same capacity: 4 frequency slots. The provisioned request set R contains r_1 and r_2, request r_1 uses a routing path from node A to node D on frequency slot index 2, and r_2 uses a routing path from node B to node E on frequency slot index 3. Figure 7(b) shows the Δ positions for both r_1 and r_2 (shifting a set of provisioned requests down as much as possible), therefore $b_\Delta(r_1) = e_\Delta(r_1) = 1$, and $b_\Delta(r_2) = e_\Delta(r_2) = 2$. Figure 7(c) illustrates the ∇ positions (shifting a set of provisioned requests up as much as possible) for both requests. $b_\nabla(r_1) = e_\nabla(r_1) = 3$ and $b_\nabla(r_2) = e_\nabla(r_2) = 4$.

The **delay** (δ) of insertion of a new request using push-pull indicates the duration of the shifting done to free the needed space. In [32], the authors take the number of slots through which the shifting is done over as an indicator of the delay and consider two types of parallelism to compute it as illustrated in Figure 8. In Figure 8(a), requests r_1 and r_2 are both shifted in the same direction (down) by two slots and one slot respectively, the delay of shifting is $\delta = \max\{2,1\} = 2$. In Figure 8(b), request r_1 is shifted by two slots and r_2 is shifted in the opposite direction by one slot. The delay of shifting is $\delta = \max\{2,1\} = 2$



(a) Example 1

(b) Example 2

Figure 8: Push-pull delay [6]

The **absolute position** is a position in frequency slot range, i.e., an index in the interval $[1,S]$ which can be assigned to r satisfying spectrum constraints in the context of RMSA. The **Relative**

position, (A,B), is defined as a position between two sets of provisioned requests. Allocating request r in a position (A,B) means that request r is above the set of requests A and below the set of requests B. (A,B) is valid if no request in B is constrained to be below a request of A. A relative position (A,B) is valid on a link ℓ if the position is valid and (A,B) is a partition of the requests using ℓ. We denote a **complete relative position** as (A_c,B_c) for every relative position (A,B). A_c contains the request set A and all the requests are constrained to be below them and B_c contains the request set B and all the requests are constrained to be above them. We say that two relative positions (A,B) and (C,D) are conflicting if and only if $A_c \cap D_c \neq \emptyset$ or $C_c \cap B_c \neq \emptyset$.

We assume that the network undergoes a series of connection re-optimization at different time units. Let T (indexed by t) be the set of those time units, with $t = 0$ being the initial one. The details of the push-pull defragmentation algorithms are discussed in the following chapter.

3.5 Problem Statement

In this subsection, we formally state the problems as follows, including both reactive and proactive defragmentation. Both reactive and proactive defragmentations include two scenarios, i.e., defragmentation with and without push-pull.

Reactive Push-Pull Defragmentation

Input:

- An optical provisioning network with a set of provisioned requests R.
- A dynamic RMSA provisioning process, i.e., a timeline (length is T) with a set of pre-defined new incoming requests and a set of departure requests in each time unit.

Provisioning Strategies:

- *Scenario 1:* For each new incoming ADD request, we use an online RMSA algorithm that searches for an available routing path (e.g., the first shortest one) which satisfies the slot requirements (contiguity and continuity constraints) and then assigns it to the new request.

In this scenario, no reactive push-pull defragmentation is introduced. Denote by R_1 the resulting RMSA provisioning.

- *Scenario 2:* For each new incoming ADD request, we use an online RMSA algorithm proposed in *Scenario 1* to search for an available position (an available routing path with a spectrum allocation which satisfies spectrum constraints) for the new request. However, if the proposed online RMSA algorithm cannot grant the new request, reactive push-pull defragmentation is triggered. The new request is denied only if there is no feasible position for it after reactive push-pull shifting. Denote by R_2 the resulting RMSA provisioning.

Output: The performance of push-pull in reactive defragmentation.

Proactive Push-Pull and Make-Before-Break Defragmentation

Input:

- An optical provisioning network with a set of provisioned requests R

Defragmentation Strategies:

- *Scenario 1:* In an online RMSA scheme, for each provisioned request r, check if there is an exact shorter available routing path that can assign r on a feasible spectrum allocation knowing that only make-before-break based defragmentation can be used.

- *Scenario 2:* In an online RMSA scheme, for each provisioned request r, check if there is an exact shorter available routing path that can assign r on a feasible spectrum allocation knowing that only make-before-break and push-pull defragmentation can be used.

Output: The performance of push-pull and make-before-break in proactive defragmentation.

Chapter 4

Reactive Push-Pull Defragmentation

In this chapter, we focus on the RMSA provisioning process with reactive defragmentation. For the RMSA planning method, *k* shortest path-based first-fit strategy has been discussed in Chapter 3, Algorithm 1. For RMSA defragmentation, two algorithms related to push-pull are introduced in this chapter. Since additional costly transponders are needed by make-before-break fragmentation [6], and reactive defragmentation is triggered as soon as a request is denied. In order to reduce the defragmentation expenses, only Push-Pull defragmentation is presented in this chapter. For each algorithm, we will present the idea of the scenario, pseudo-code, and running time complexity analysis.

4.1 Introduction

In an online RMSA scenario, requests with different data rates are allocated and disconnected in a quite random fashion. In the scenario without any defragmentation, spectral resources tend to be highly fragmented because of the spectrum constraints [28]. In order to improve network fragmentation and to grant more requests, in this chapter, we introduce reactive Push-Pull defragmentation in online RMSA.

Reactive push-pull defragmentation is triggered when a new incoming request is denied. In order to grant the request on a feasible spectrum allocation (which satisfies spectrum continuous and contiguous constraints) with the minimum push-pull shifting delay, the following two problems need to be investigated.

1. How to find a set of links which are able to make enough space for the denied request r under RMSA scenario knowing that only push-pull can be used? Denote as **RMSA-PP** problem.

2. How to provision a request r (with a given routing path) using a minimal defragmentation delay while using push-pull? Denote as **SA-PP** problem.

We formally state the problems as follows:

Problem 1 (RMSA-PP). Given an optical provisioning network, a set of provisioned requests R, and a new request r with data rate, source SRC_r and destination DST_r, is it possible to find a set of links that have routes from source to destination with enough space for r (unknown modulation), knowing that only push-pull can be used? (The algorithm is discussed in Section 4.2)

Problem 2 (SA-PP). Given an optical provisioning network, a set of provisioned requests R, and a new incoming request r with slot demands d_r^m, a modulation and routing path p_r, is it possible to assign spectrum interval to r, with minimum shifting delay, knowing that only push-pull can be used? (The algorithm is discussed in Section 4.3)

4.2 Routing, Modulation and Spectrum Assignment with Push-Pull (RMSA-PP)

Routing and Spectrum Assignment with push-pull (RSA-PP) was solved in [6]. The only difference between RMSA-PP and RSA-PP is that modulation in RSA-PP is fixed, in other words, slot requirements in RSA-PP only depends on request data rates. In this chapter, we will modify the RSA-PP algorithm in [6] and use it to solve RMSA-PP.

Algorithm 2 solves the RMSA-PP problem by finding the available links on each spectrum allocation. The idea is as follows: Suppose r is a new incoming request from source SRC_r to destination DST_r with its data rate. Since the modulation is not set yet, the algorithm will try all the modulation formats associated with request demands d_r^m (slot demands of request r under modulation m), the feasibility of the modulation m will be checked later (a feasible modulation means that the length of the routing path matches the modulation distance requirements). For each d_r^m, there are at most $S - d_r^m + 1$ absolute positions to allocate request r. For each absolute position λ, we create a graph consisting of a set of links L_λ, each link $\ell_\lambda \in L_\lambda$ is able to make enough space for request r by using push-pull defragmentation on spectrum allocation λ, we find the shortest path from the source SRC_r to the destination DST_r in the graph. And then, we choose the shortest feasible path among the ones found for each λ, the feasible path means that the length of the routing path is modulation m reachable. In the end, we choose the one with the shortest geographical distance among the paths we found for each modulation.

The feasibility checking on each link ℓ is based on **Lemma 1** and it has been proved in [6]. Suppose that a new request r is provisioned, r will partition the confliction set $CS(r)$ into two subsets: requests above r and requests below r. A partition $(A, A'$ gives the largest free interval when we shift the requests that are above (i.e. A) to the ∇ state and those below (i.e., A') to Δ state). The widths of the position $\alpha = A, A'$ before and after defragmentation are given by $w(\alpha) = c(\alpha) - f(\alpha)$ and $w^\star(\alpha) = c^\star(\alpha) - f^\star(\alpha)$. Therefore, in order to check if we can provision request r at a position $\alpha = (A, A')$ on link ℓ, it is enough to check if $w^\star(\alpha) \geq d_r^m$.

In addition, the feasibility checking the non-confliction shiftings on an absolute position is based on **Lemma 2**. It has been proved in [6] that if we can free a position λ on each of links ℓ_λ in L_λ, i.e., freeing all frequency slots indexed in $[\lambda, \lambda + d_r^m - 1]$, then we can free this position on L_λ using non-conflicting shiftings.

Lemma 1: *Let r be a request of demand d_r and path p_r, $|CS(p_r)| = k$ and α_i, where $i \in 0, ..., k$ are the corresponding decision-positions. Request r is provisionable over p_r using push-pull, if and only if there exists some $i \in \{0, ..., k\}$, such that $\omega^\star(\alpha_i) \geq d_r^m$*

Lemma 2: *If the absolute position λ can be freed for a request r with demand d_r^m on a set of links E, then there are valid non-conflicting relative positions on the links of E which can free λ for r.*

Algorithm 2: RMSA-PP(r, R, Mod) $\rightarrow p_r$, shortest available path selection

```
   Input  : Optical provisioning network, a set of modulation format Mod, the provisioned
            request set $R$ and a new incoming request $r$ with its data rate, source $\mathrm{SRC}_r$ and
            destination $\mathrm{DST}_r$
   Output: Shortest available routing path for $r$
1  Sort Mod in the decreasing order of modulation level;
2  Initialize $p_r = \emptyset$;
3  for ( $m \in Mod$ ) {
4      Set request slot requirements $d_r^m$ base on $m$;
5      for ( $\ell \in L$ ) {
6          Sort provisioned requests which are using $\ell$ in the increasing order of $b(r)$, denote
             the sorting list as $< r_1, ...., r_k >$;
7          initialize $e_{\triangle}(r_0) = 0$; and $b_{\bigtriangledown}(r_{k+1}) = |S|$;
8          for ( $\lambda \in S$ ) {
9              for ( $i \in \{0, 1, ..., k\}$ ) {
10                 if $[\lambda, \lambda + d_r^m - 1] \subseteq [e_{\triangle}(r_i), b_{\bigtriangledown}(r_{i+1})]$ then
11                     color link $\ell$ with color $\lambda_m$ and break;
12     Find the shortest mono-colored $st$-path $p$, (a monocolored path is a path whose links
         share a color);
13     if the length of p is modulation m reachable then
14         modulation of $r = m$;
15         $d_r = d_r^m$;
16         $p_r = p$;
17         return $p_r$;
```

Here is the algorithm explanation and running time complexity analysis:

Step 3 sorts all the modulations Mod. **Step 4** assumes that r is using modulation m and the slot demands is set based on m, the feasibility of m will be checked later. **Step 5** selects all the possible link ℓ in the optical provisioning network. **Step 6** sorts all the requests using link ℓ, the running time for sorting is $O(|R'| \times \log(|R'|))$, where R' is a set of requests using link ℓ, in the worst case, sorting operation invokes all the provisioned requests. **Step 8** is a loop for checking the feasibilities for spectrum allocations and the number of iterations is bounded by $|S|$. **Step 9** checks all the possible relative positions in set R'. **Step 10-11** check the feasibility of relative locations according to **Lemma 1**. **Step 12** finds the shortest path using Fredman's alogorithm [13], which takes $O(L+N\log N)$, where N and L are nodes and links in the provisioning network respectively. **Step 13** checks the feasibility of the modulation m, if the length of p is reachable for the modulation distance requirements, then we set r's modulation format as m, routing path as p and return, otherwise, continue.

The worst time complexity for RMSA-PP is $O(\mathrm{M}(L(|\mathrm{R}|\log|\mathrm{R}|+|S||\mathrm{R}|)+N\log N+\log\mathrm{M}))$. where M is the number of modulations, N and L are the number of nodes and links in the optical provisioning network, |S| is the link capacity and |R| is the number of the provisioned request. In this thesis, modulation format only contains BPSK, QPSK, 8QAM and 16QAM, and the link capacity S is a constant (400 slots), therefore, we can simplify the time complexity for RMSA-PP as $O(L|\mathrm{R}|\log|\mathrm{R}|+N\log N)$.

4.3 Spectrum Assignment with Push-Pull (SA-PP)

It has been proven in [33], that SA-PP can be solved in polynomial time. Authors of [6] proposed an algorithm to find the spectrum allocation with the minimum delay on a given routing path. The idea is as follows: In order to find a location with the minimum delay, we need to know how many extra spaces are needed in a given partition α to grant r, and push-pull operation stops as soon as the required rooms are freed.

Here is the analysis to calculate the delay of a given position in push-pull technique:

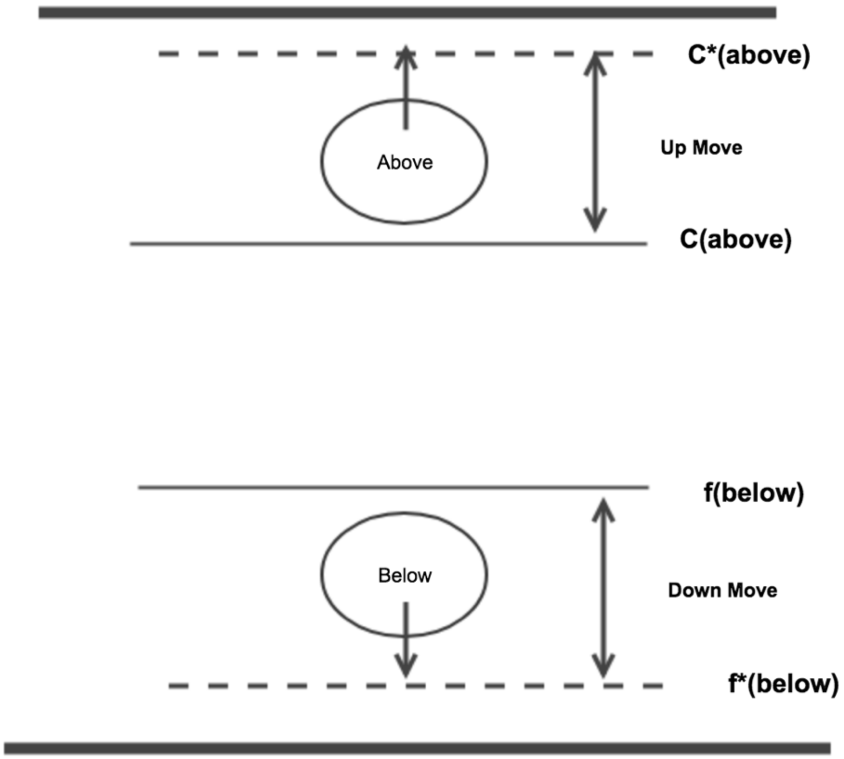


Figure 9: Push-pull delay calculation

As shown in Figure 9, suppose that the request demand is d_r, given a partition $\alpha = (A, A')$. The original width of this partition is $\omega(\alpha) = c(\alpha) - f(\alpha)$. When the push-pull is triggered, we push A' up and pull A down at the same time. Push-Pull operation stops if $d_r - \omega(\alpha) = shifts(A) + shifts(A')$, where $shifts(A) = f(\alpha) - f^{\star}(\alpha)$, $shifts(A') = c(\alpha^{\star}) - c(\alpha)$ and $d_r - \omega(\alpha)$ is the extra needed spaces in order to grant r. The following 3 cases are considered, we define delay of partition α as $\delta(\alpha)$:

Case1 : If $shifts(A) > shifts(A')$, then $\delta(\alpha) = shifts(A)$ and $\delta(\alpha) = d_r - \omega(\alpha) - shift(A')$, therefore, $\delta(\alpha) = d_r - \omega(\alpha) - (f(\alpha) - f^{\star}(\alpha))$.

Case2 : If $shifts(A) < shifts(A')$, then $\delta(\alpha) = shift(A')$. Therefore, $\delta(\alpha) = d_r - \omega(\alpha) - (c^{\star}(\alpha) - c(\alpha))$.

Case3 : If $shift(A) = shift(A')$. In this case $\delta(\alpha) = d_r - \omega(\alpha) - \left\lfloor \frac{d_r - \omega(\alpha)}{2} \right\rfloor$.

In summary, we can write the delay function as follows:

$$\delta(\alpha) = d_r - \omega(\alpha) - \min\{f(\alpha) - f^\star(\alpha), c^\star(\alpha) - c(\alpha), \left\lfloor \frac{d_r - \omega(\alpha)}{2} \right\rfloor\}.$$

In order to allocate the request r with the minimum delay, let $CS(r)$ be sorted as $< r_1, r_2, ..., r_k >$ in the ascending order of the requests ending slots in the $\triangle$ state, (recalling that $\triangle$ state is to shift requests down until they are blocked), i.e., $e_\triangle(r_1) \leq e_\triangle(r_2) \leq \cdots \leq e_\triangle(r_k)$. The defined *decision-positions* of r over path p_r as the $k+1$ positions α_i such that $\alpha_0 = (\emptyset, r_1, ..., r_k)$ and $\alpha_i = (r_1, ..., r_i, r_{i+1}, ..., r_k)$, for $i \in [0,k]$. It has been proved in [33] that to decide if it is possible to assign a spectrum set to r on path p_r, it is sufficient to check the $k+1$ decision-positions α_i, $i \in 0, ..., k$.

Thanks to **Lemma 1**, to solve the SA-PP problem, the authors in [33] and [6] have designed an algorithm which checks all of the decision-positions then chooses the one over which the new request can be provisioned with minimum delay. The pseudo-code of the proposed algorithm is presented in Algorithm 3.

Algorithm 3: SA-PP(r, R) $\rightarrow \lambda$, spectrum allocation with the minimum delay

```
Input  : Optical provisioning network, a set of provisioned requests $R$ and a new incoming
         request $r$ with its demand $d_r$ and routing path $p_r$
Output: Allocate spectrum to $r$ on a spectrum domain with the minimum delay
Initializing $\lambda = Null$, $\beta = \emptyset$ and $\delta = \infty$
Find $CS(r)$ the set of requests conflicting with $r$ and sort it in the ascending order of $e_{\triangle}$.
  Denote the sorted list as $< r_1, r_2, ..., r_k >$. The corresponding decision-positions are
  $\alpha_0, \alpha_1, ...\alpha_n$, such that $\alpha_i = (A_i, A'_i)$
for ( $i \in \{0, 1, ...n\}$ ) {
    if $\omega^*(\alpha_i) \geq d_r$ then
        Sort the requests in $A_i$ in the descending order of $e(x)$. The sorted list is
          $< x_1, x_2, ..., x_i >$ and $\alpha_i^j = \left(\left\{A_i \setminus \{x_1, x_2, ..., x_j\}\right\} \cup \left\{\overline{A}_i, \{x_1, x_2, ..., x_j\}\right\}\right)$ where
          $j \in \{1, ..., i-1\}$ and $\alpha_i^0 = \alpha_i$
        for ( $j \in \{0, ..., i-1\}$ ) {
            if $\delta\left(\alpha_i^j\right) < \delta$ then
                $\beta = \alpha_i^j$ and $\delta = \delta\left(\alpha_i^j\right)$ and $\lambda = \max\left(f(\beta) - \delta, f^*(\beta)\right)$
```

Here is the algorithm explanation and running time complexity analysis:

Step 1 is a setting operation, where β is the desired position, initialized with NULL. **Step 2** finds the conflicting set of r, $CS(r)$, and sorts it. Taking $O(|R'|\log|R'|)$ time complexity, where $|R'| = |CS(r)|$ and is bounded by $|R|$. **Step 3** iterates the possible decision positions. In the worst case, this step invokes all the provisioned requests. **Step 4** calculates the largest spectrum interval according to a partition α_i. **Step 5** sorts the requests below the partition, in the worst case, the time complexity of this step is $O(|R|\log|R|)$. **Step 6-8** checks all the feasible relative positions in order to find the minimum delay, in the worst case, this step invokes all the provisioned requests.

Therefore, the overall time complexity of Algorithm 3 is $O(|R|^2\log|R|)$, where $|R|$ is the number of the provisioned requests.

Chapter 5

Proactive Push-Pull and Make-Before-Break Defragmentation

In this chapter, we focus on proactive defragmentation. It is performed periodically or according to some fragmentation degradation thresholds in order to maintain spectral defragmentation at an acceptable level. In addition, proactive defragmentation is independent of request connection events [27]. Since proactive defragmentation is performed periodically and push-pull defragmentation introduces delays. In order to reduce the defragmentation delay, we will combine make-before-break and push-pull in the proactive defragmentation. We will compare two proactive defragmentation techniques, i.e., make-before-break (or MBB for short), and make-before-break combined with push-pull (or MBBPP for short). We will then define them.

Proactive MBB. Given an optical provisioning network with a set of requests *R*, what is the most efficient way to use Make-Before-Break in order to reduce the spectrum usage as much as possible? See our proposal in Section 5.1.

Proactive MBBPP. Given an optical provisioning network with a set of provisioned requests *R*, what is the most efficient way to use a combination of Push-Pull and Make-Before-Break in order to reduce the spectrum usage as much as possible? See our proposal in Section 5.2.

In the sequel, we define spectrum usage in a provisioned network as follows:

$$SU = \sum_{r \in R} \sum_{\ell \in p_r} w_\ell \, d_r,$$

where w_ℓ is the distance/length of link ℓ and d_r is the slot demand of r.

5.1 Proactive Defragmentation with MBB

In this section, we discuss proactive defragmentation with make-before-break (MBB). The objective is to design an algorithm to re-configure provisioned requests with a reduced spectrum usage.

The idea is to grant requests on the shortest available routing path. A provisioned request that uses a shorter route means that there is less spectrum usage [12]. In addition, a shorter routing path may change the request modulation from a lower modulation level to a higher level one, and a higher level of modulation means a higher spectrum efficiency. Spectrum efficiency is the ratio of bit rate to available bandwidth, therefore, a higher level of spectrum efficiency reduces the slot requirements to carry the same data rate requests, which has been discussed in Section 3.2.

Another concern that is addressed in this thesis, is the online RMSA environment, therefore, the defragmentation algorithm needs to be simple and fast [27]. Therefore, the proposed algorithm only considers those requests not using the shortest path and it is described in Algorithm 4.

Algorithm 4: MBB(R)→ R', proactive defragmentation using only make-before-break

```
   Input  : An optical provisioning network with a set of requests $R$
   Output: A new request provisioning $R'$ with a reduced spectrum usage
1  Initialize $R' = R$;
2  Sort the requests of $R'$ in the decreasing order of their slot demands;
3  for ( $r \in R'$ : r is not on its shortest possible path ) {
4      KSHP = Set of $k$ shortest paths of $r$ from $\mathrm{SRC}_r$ to $\mathrm{DST}_r$;
5      for ( $p \in KSHP$ ) {
6          if $\text{LENGTH}(p) \geq \text{LENGTH}(p_r)$ then
7              Go to step 3;
8          $d_{temp}$ = slot demand based on a modulation $m$ ($m$ is the highest-level modulation
             associated with routing path $p$'s distance);
9          for ( $s \in S$ ) {
10             if $[s, s + d_{temp} - 1]$ is feasible on routing path $p$ then
11                 set $p_r$ = $p$; set $b(r)$ = $s$;
12                 set $r$'s modulation as $m$; set $d_r$ = $d_{temp}$;
13                 Go to step 3;
14 Return $R'$
```

Here is the algorithm explanation and running time complexity analysis:

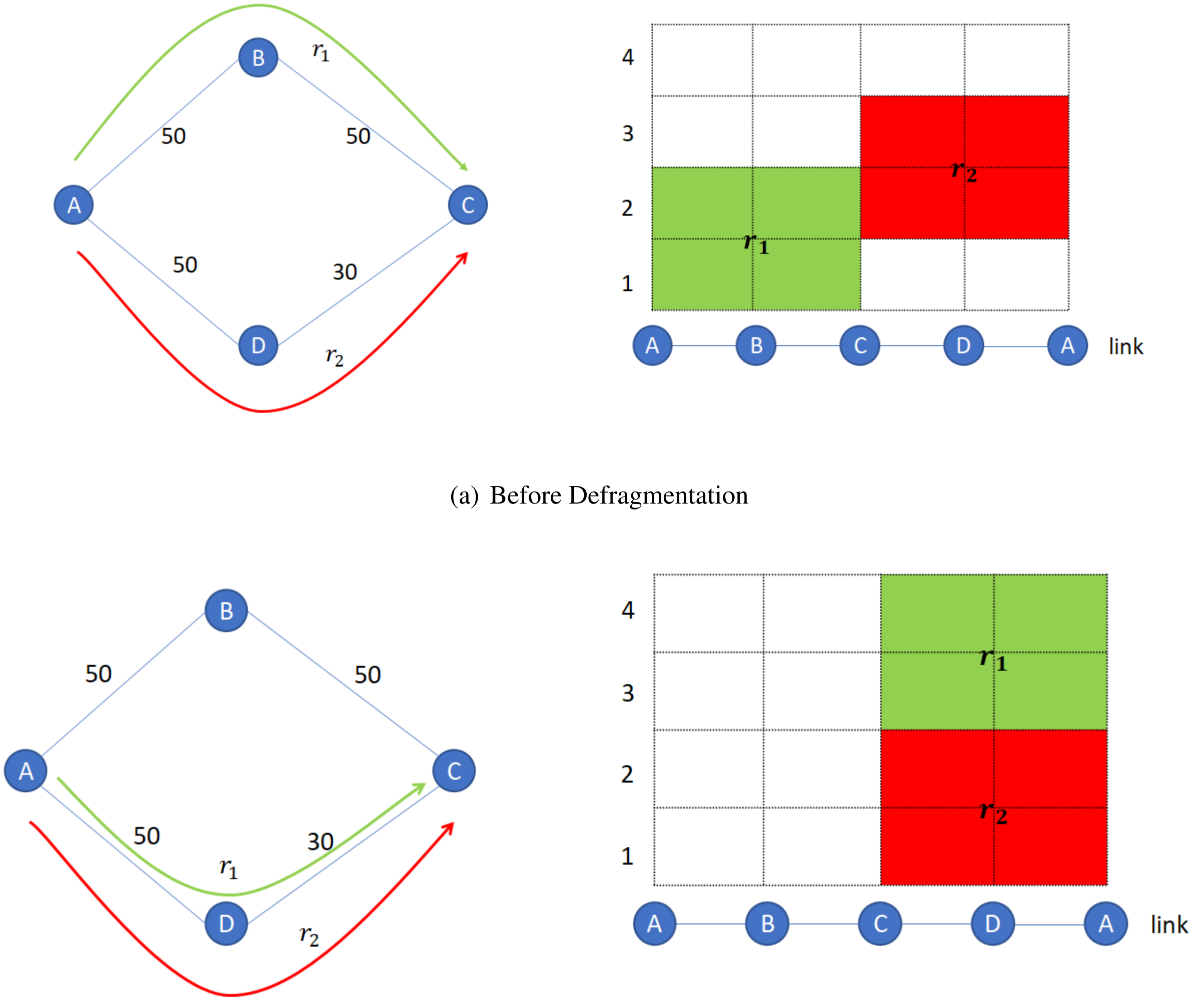
Step 2 sorts the provisioned requests R, takes $O(|R| \log |R|)$. **Step 3** iterates all the provisioned requests which are not routed on one of their shortest paths. **Step 4** finds the k-shortest path using Yen's algorithm [38]. The running time for this algorithm is $O(kN(L + N \log N)$. **Step 5** iterates over all the candidate routing paths. **Step 8** selects frequency slots according to a modulation format. The highest-level modulation can be used on the selected path p. **Step 9** loops over all the set of frequency slots, $O(|S|)$ time complexity. **Step 10-14** check if spectrum allocation s is available using the candidate routing path and slot demands. The time complexity of step 10 is dependent on the size of the path and slot demands.

Therefore, the overall running time complexity is $O(|R|\log|R|+k|R|(NL+N^2\log N+|S|^2L))$. Since the link capacity S is a constant (400 slots) and k is a small integer, we choose $k=3$ in the thesis. Therefore, we simplify the time complexity as $O(|R|(\log|R|+NL+N^2\log N))$

5.2 Proactive Defragmentation with MBB and Push-Pull

In the previous section, we proposed a proactive defragmentation algorithm based on make-before-break. However, only make-before-break does not always work as we will see in the following example. As shown in Figure 10, a network topology has 4 nodes (A,B,C,D) and 4 arcs, i.e., A-B, B-C, C-D, and D-A. r_1 and r_2 are 2 provisioned requests, where r_1 is from node A to node C, uses routing path A-B-C and is assigned slots 1 and 2; r_2 is from node A to node C, uses routing path A-D-C and is assigned slots 2 and 3. Make-Before-Break technique cannot reconfigure r_1 on routing paths A-D-C (if r_2's provisioning is not changed), which has a shorter routing distance, because of the spectrum contiguity constraints.

In order to overcome such an issue, we combine the push-pull and make-before-break techniques to improve the proactive defragmentation performance. Request r_1 cannot be rerouted on path A-D-C because of the spectrum constraints. In order to reroute r_1 on a better routing path, we trigger push-pull. As a result, r_2 can be pulled down (or pushed up) and be assigned lower (or higher) indexed slots using the push-pull technique. Figure 10(b) illustrates the network provisioning after defragmentation. Request r_2 is re-assigned slots 1 and 2 using push-pull and request r_1 is rerouted on paths A-D-C and is re-assigned slots 3 and 4 using make-before-break.



(a) Before Defragmentation

(b) After Defragmnetation

Figure 10: MBBPP example

The idea is as follows: Consider the provisioned requests which are not provisioned on their shortest path. Try Algorithm 4 to re-provision the requests which are not provisioned on the shortest path. If only make-before-break cannot find a better provisioning (make-before-break with no changing other provisioned requests), we trigger push-pull defragmentation, the routing path selection uses RMSA-PP (Algorithm 2) and spectrum allocation selection uses SA-PP (Algorithm 3). The resulting algorithm is described in Algorithm 5.

Algorithm 5: MBBPP(R) $\rightarrow R'$, Proactive defragmentation with make-before-break and push-pull

```
   Input  : A provisioning network with a set of established requests $R$
   Output: A new request provisioning $R'$, which has a reduced spectrum usage
 1 Initialize $R' = R$;
 2 Sort $R'$ in decreasing order of request demands;
 3 for ( $r \in R' \setminus \{r' | r' \in R' \ \& \ r' \textit{ on the shortest path} \}$ ) {
 4     $p_{av}$ = RMSA-PP($r$, $R' \setminus \{r\}$, Mod);
 5     KSHP = Find $k$ shortest path from SRC$_r$ to DST$_r$;
 6     for ( $p \in \{p' | p' \in KSHP \ \& \ p' < p_{av} \ \& \ p' < p_r\}$ ) {
 7         $d_{temp}$ = slot demand associated with $p$ and a modulation $m$;
 8         for ( $s \in S$ ) {
 9             if $[s, s + d_{temp} - 1]$ is feasible on routing path $p$ then
10                 set $p_r$=$p$; set $b(r)$=$s$;
11                 set $r$'s modulation as $m$;
12                 Go to step 3;
13     if $p_{av} < p_r$ then
14         $\lambda$ = SA-PP ($r$, $R' \setminus \{r\}$);
15         set $p_r$=$p_{av}$; set $b(r)$=$\lambda$;
16         set $r$'s modulation which is obtained from RMSA-PP;
17 Return $R'$
```

Here is the algorithm explanation and running time complexity analysis:

Step 2 sorts the provisioned requests R, $O(|R| \log |R|)$. **Step 3** iterates all the provisioned requests which are not routed on one of their shortest paths. **Step 4** finds the shortest available routing path in provisioning network by using Algorithm 2 proposed in Chapter 4. Algorithm 2 takes $O(L|\mathrm{R}| \log |\mathrm{R}| + N \log N)$ time complexity. **Step 5**, finds the k shortest path using Yen's algorithm

[38] and takes $O(kN(L+N\log N)$ computational complexity. **Step 6-12**, are the steps for make-before-break technique. The length of the candidate routing paths are less than the original one and the shortest available path p_{av}. The loop has a time complexity of $O(k|S|^2L)$; **Step 13-16**, if make-before-break is not able to reconfigure the request and the shortest available routing path is less than the original one, use Algorithm 3 to get the spectrum allocation with the minimum delay.

The overall computation complexity is $O(|R|(L|R|\log|R|+N\log N+kN(L+N\log N)+k|S|^2L+|R|^2\log|R|)$. As we have discussed in the previous section, the link capacity S is a constant and k is a small integer. Therefore, in this thesis, the running time complexity is $O(|R|(L|R|\log|R|+NL+N^2\log N+|R|^2\log|R|))$.

Chapter 6

Numerical Results

This chapter mainly discusses the dynamic RMSA defragmentation on different topologies using push-pull with various combinations of reactive and proactive strategies. Firstly, we will describe the details of the data sets in Section 6.1.1, then, of the simulation environment, i.e., RMSA provisioning strategies and defragmentation triggering events in Section 6.1.2. We conducted different numerical analysis, which are reported in the remaining subsections.

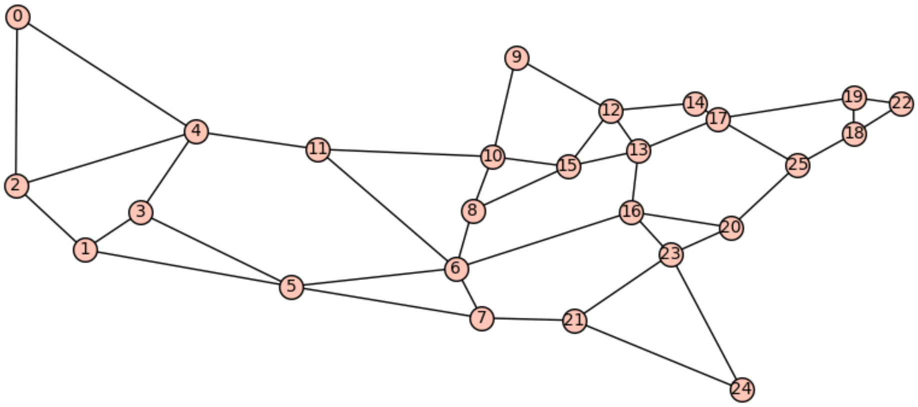
6.1 Experiment Framework

6.1.1 Data Sets

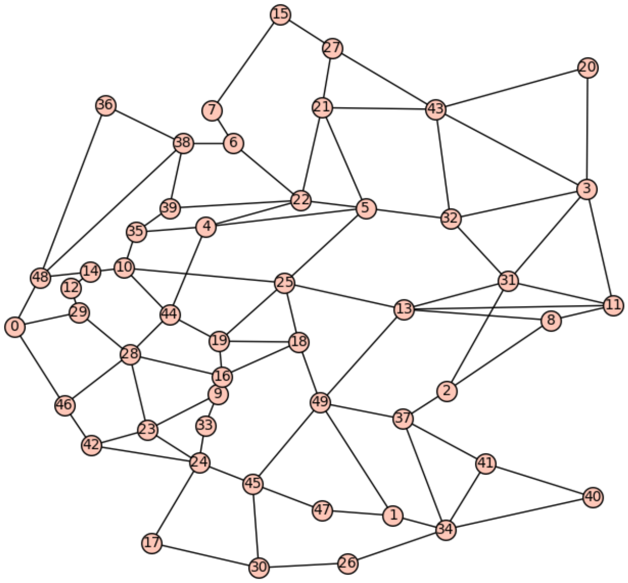
Datasets are from **SNDlib** [21], **Monarch Network Architects** [20] and **The Internet Topology Zoo** [17]. We run our simulations on 4 different topologies, namely USA (shown in Figure 11(a)), Germany (shown in Figure 11(b)), CONUS (shown in Figure 11(c)) and NTT (shown in Figure 11(d)). The key characteristics of each topology (number of nodes and links, average node degrees) are described in Table 1. The simulation runs on directed topologies.

Networks	$\|V\|$	$\|L\|$	Avg.deg	Di-Gragh
USA	24	88	3.7	True
Germany	50	176	3.5	
CONUS	60	158	2.6	
NTT	55	144	2.6	

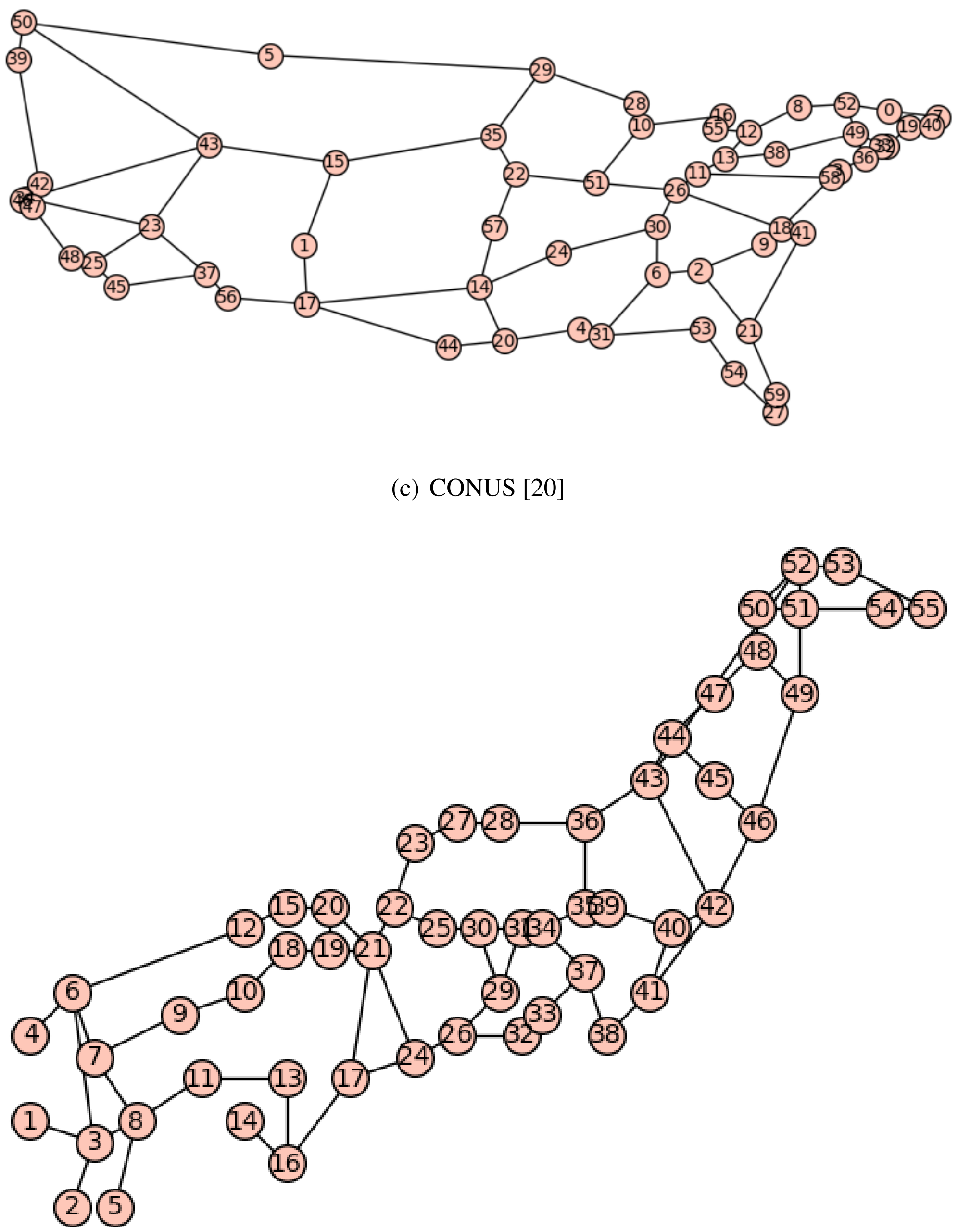
Table 1: Main characteristics of the networks



(a) USA [21]



(b) Germany [21]



(c) CONUS [20]

(d) NTT [17]

Figure 11: Network topologies

6.1.2 Defragmentation Strategies: Triggering Events

The spectrum for each link is slotted into 400 slots where each slot corresponds to a spectrum interval of width 12.5 GHz. The simulation starts from a loaded optical network. The provisioning sequence of requests arriving and departing is predefined, specifically, requests arrive and depart in the network with a **Poisson Distribution**, $P(k) = e^{-\lambda}\frac{\lambda^k}{k!}$, with $\lambda_a = \lambda_d = 1$, where λ_a is the request arrival rate and λ_d is the request departure rate, which means that in a time unit, there is an average of 1 request arriving and 1 request departing. For each topology, a total of at least 5×10^4 time units are simulated, according to network topology convergence statuses, the total simulated time units are different. The source and destination are randomly selected between the nodes of the network. Three types of request data rate are considered: 100 Gbps, 200 Gbps and 400 Gbps, all the invoked data rates are equally distributed . The number of slots usage with different modulation format and the data rate is given in the following Table 2. Recalling that proactive defragmentation is triggered when network status reaches a predefined threshold, which is independent on request connection events. The reactive defragmentation, on the other hand, is triggered when a request is denied.

Data Rate (Gbps)	Modulation	#Slots	Distance (km)
100	BPSK	8	>4000
	QPSK	3	4000
	8QAM	2	1200
	16QAM	1	600
200	BPSK	16	>4000
	QPSK	6	4000
	8QAM	4	1200
	16QAM	3	600
400	BPSK	32	>4000
	QPSK	12	4000
	8QAM	8	1200
	16QAM	6	600

Table 2: Table of modulation

For each topology, we compared 6 different RMSA requests provisioning and defragmentation strategies, we denote:

- **FF**: RMSA online provisioning strategy uses first-fit without any defragmentation.
- **FFPP**: RMSA online provisioning strategy uses first-fit with reactive push-pull and without proactive defragmentation.
- **FF_MBB**: RMSA online provisioning strategy uses first-fit and proactive defragmentation uses only make-before-break.
- **FFPP_MBB**: RMSA online provisioning strategy uses first-fit with reactive push-pull defragmentation. Proactive defragmentation uses only make-before-break.

- **FF_MBBPP**: RMSA online provisioning strategy uses first-fit and proactive defragmentation uses make-before-break with push-pull.
- **FFPP_MBBPP**: RMSA online provisioning strategy uses first-fit and reactive push-pull defragmentation. Proactive defragmentation uses make-before-break and push-pull.

In addition, we capture the following measurements in our simulations throughout this chapter:

- Overall Throughput **(OTH)**, which equals the sum of the data rate in the provisioned requests set , **OTH** = $\sum_r^R r_{dr}$ where R is the provisioned request set and r_{dr} is the data rate of r.
- Blocking Ratio **(BR)**, which equals to the ratio of the number of the blocked requests to the recent 1000 arriving requests in the simulation, **BR** = $num(D)/1000$, where $num(D)$ is the number of denied requests in recent 1000 request arriving.
- For each proactive defragmentation, we capture the improved values of spectrum usage $\triangle SU$, where spectrum usage SU is defined as $SU = \sum_{r\in R} \sum_{\ell \in p_r} w_\ell d_r$, where w_ℓ is the weight of link ℓ, p_r is the routing path of request r and R is the provisioned request set. The improved value $\triangle SU$ is the difference of the spectrum usage before and after proactive defragmentation.
- We track the defragmentation delays in both reactive and proactive defragmentations. Specifically, in reactive defragmentations, i.e., FFPP, FFPP_MBB, and FFPP_MBBPP scenarios, we capture the minimum, maximum and average defragmentation delay, whereas, in proactive push-pull defragmentations, namely FF_MBBPP and FFPP_MBBPP, we measure the minimum, maximum and average summed delay. The summed delay is defined as the total defragmentation delay in proactive defragmentations.

6.2 Impact of the Initial Solution

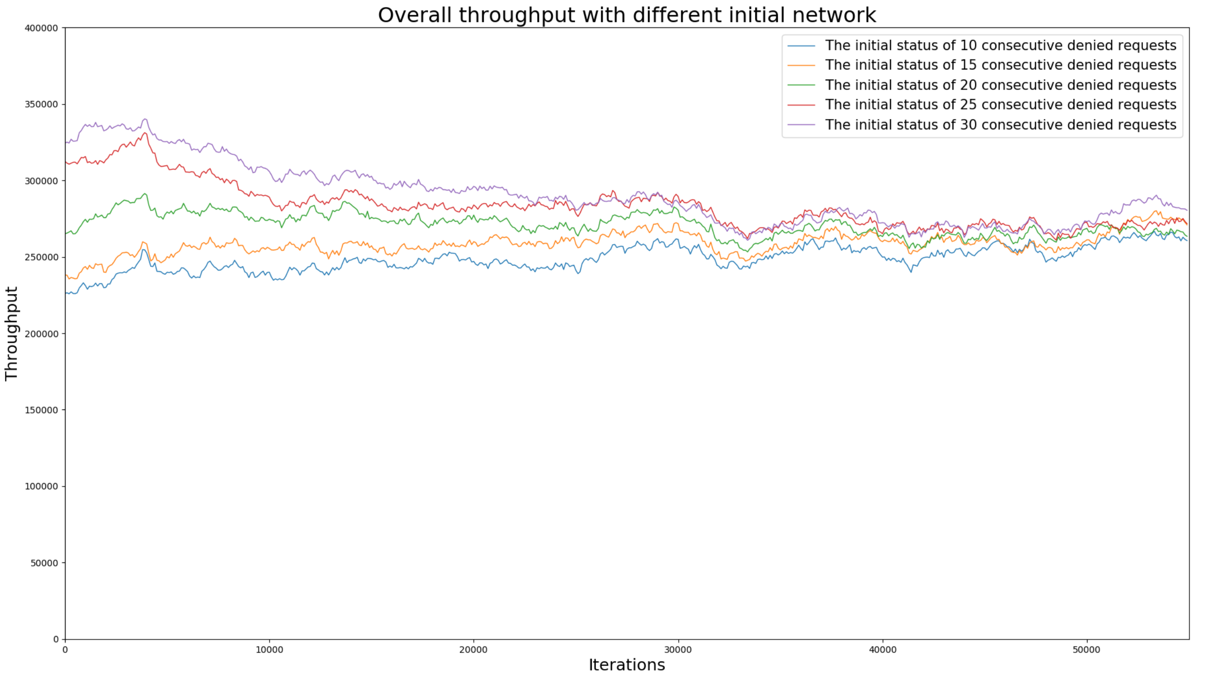
In this section, we discuss the impact of the initial network status. The initial network is generated by the following strategy. Given a positive integer i, using the Algorithm 1 (proposed in Chapter

2) to feed the network, if requests are denied consecutively up to the given integer *i*, we stop feeding and consider the network provisioning as the initial network provisioning. We choose one particular RMSA strategy, i.e., FFPP_MBBPP (we will see in the subsequent experiments that it is one of the best) and set *i* equal to 10, 15, 20, 25 and 30 respectively, and we measure the impact of the different initial network provisioning on the performance of the network overall throughput.

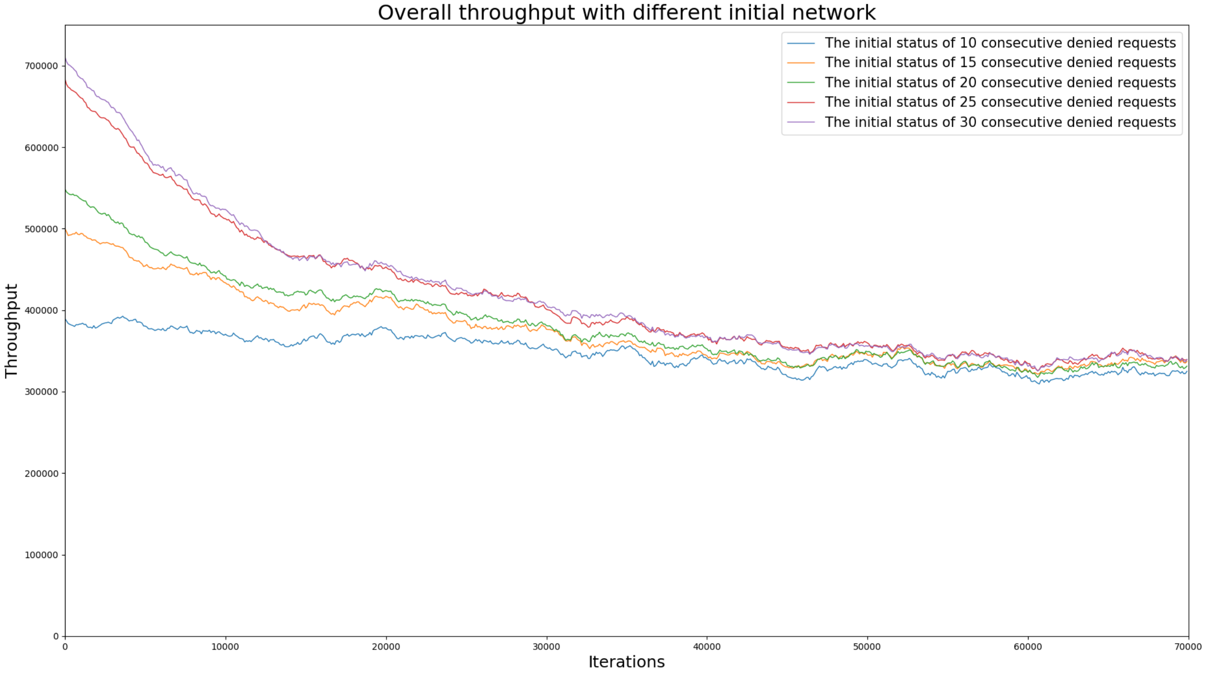
Figure 12 illustrates the impact of the different initial network provisioning on the overall throughput. The X-axis represents the simulated time units and the Y-axis is the overall throughput. As we can observe from figures below, in the same network topology, network throughputs of different initial network provisioning converge at a similar level. Therefore, different initial network provisioning do not have a significant influence on the RMSA provisioning and defragmentation strategies discussed in this thesis.

However, a larger stopping condition value, *i*, has a higher bias on requests which are using long routing hops. This is shown in Figure 13, which illustrates path distributions of each initial network condition. The bar charts show the number of provisioned requests on different hops whereas the line charts illustrate the cumulative percentage of path hops. We can observe from Figure 13, as the stop condition value *i* increases, provisioning network tends to grant more requests (according to cumulative percentage) use shorter hops.

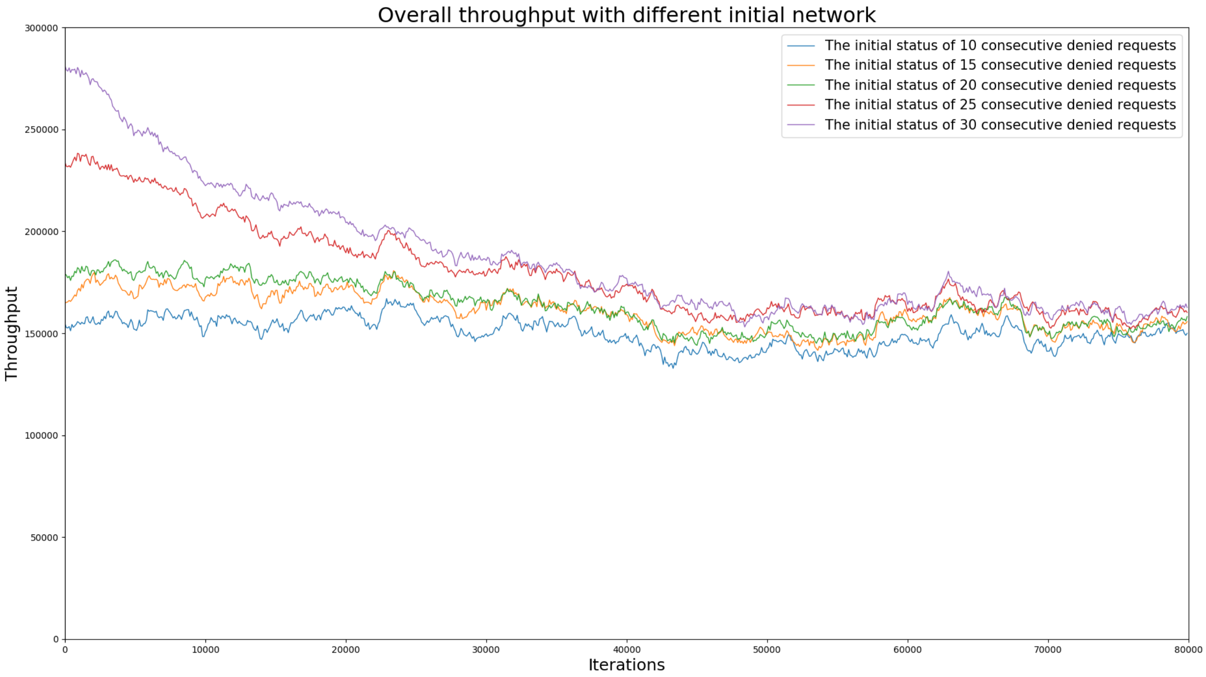
In this thesis, we choose stop condition $i = 10$, since the network status will converge faster compared with other stopping conditions and the provisioned request set has a lower hop bias.



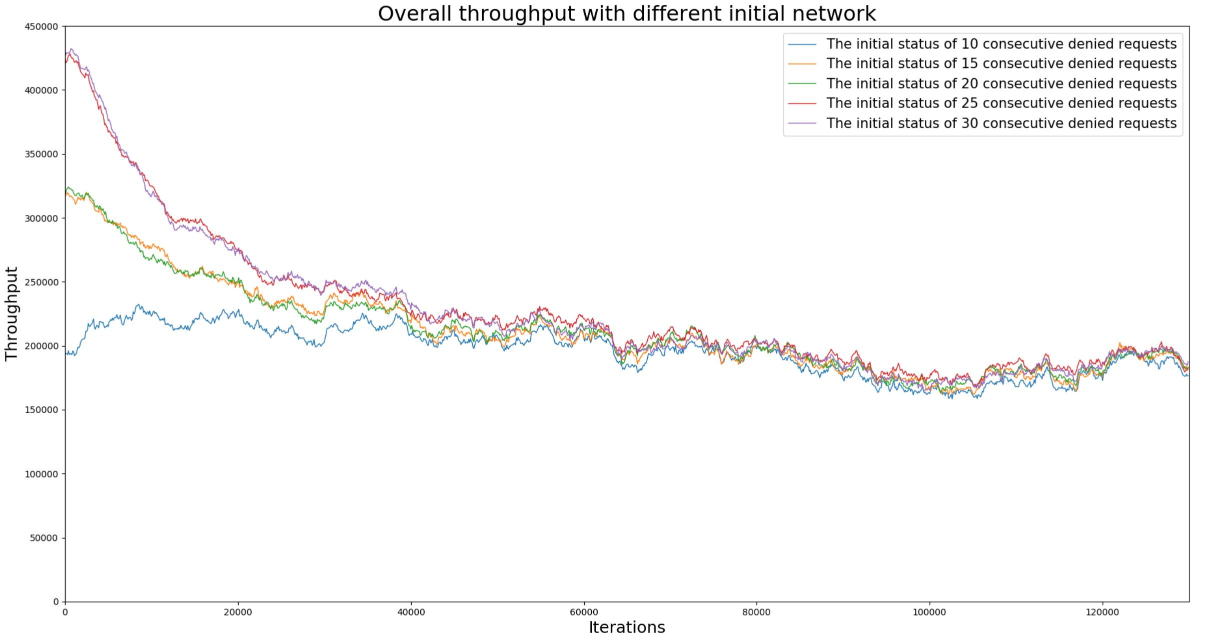
(a) USA



(b) Germany

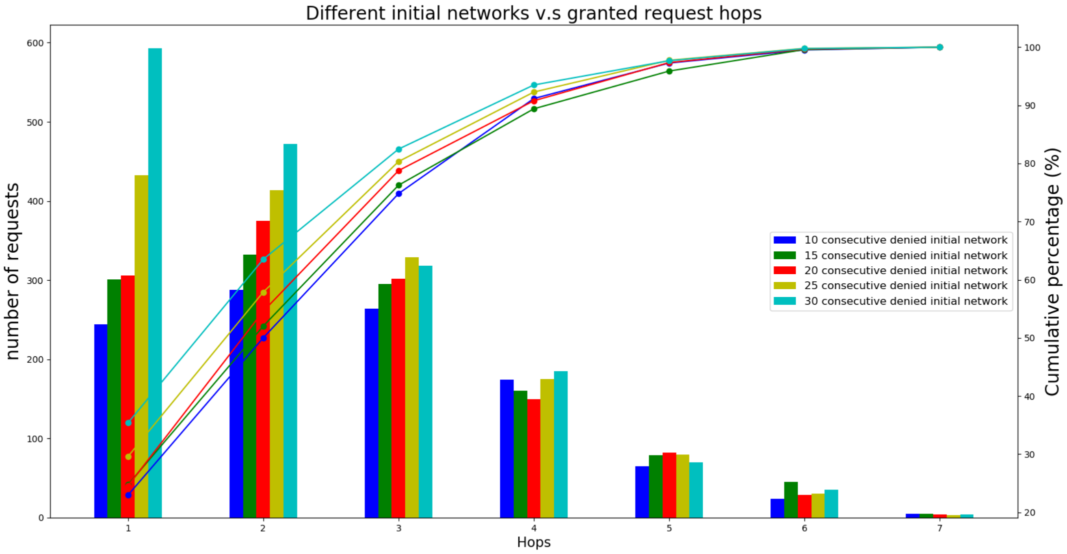


(c) CONUS

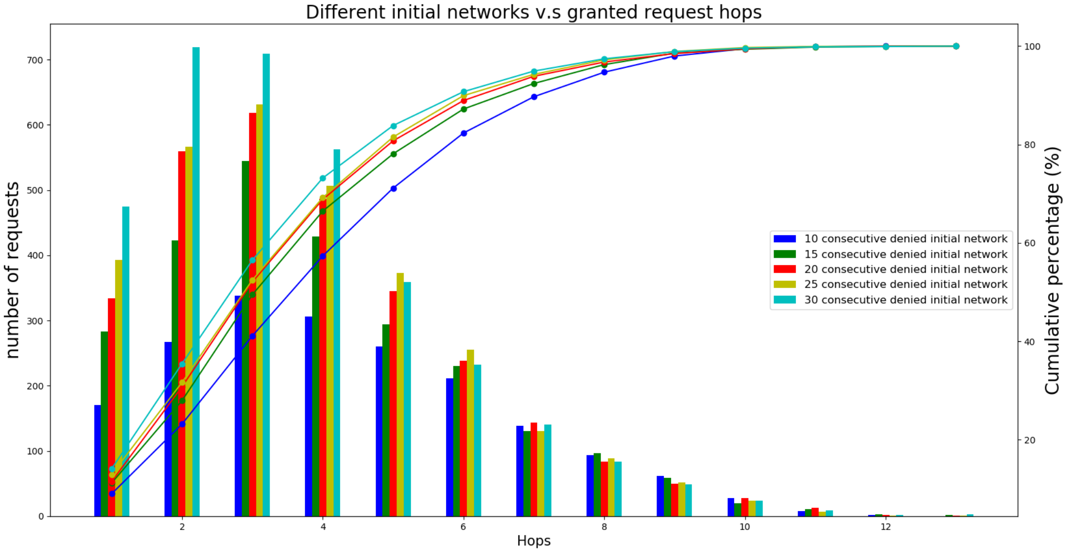


(d) NTT

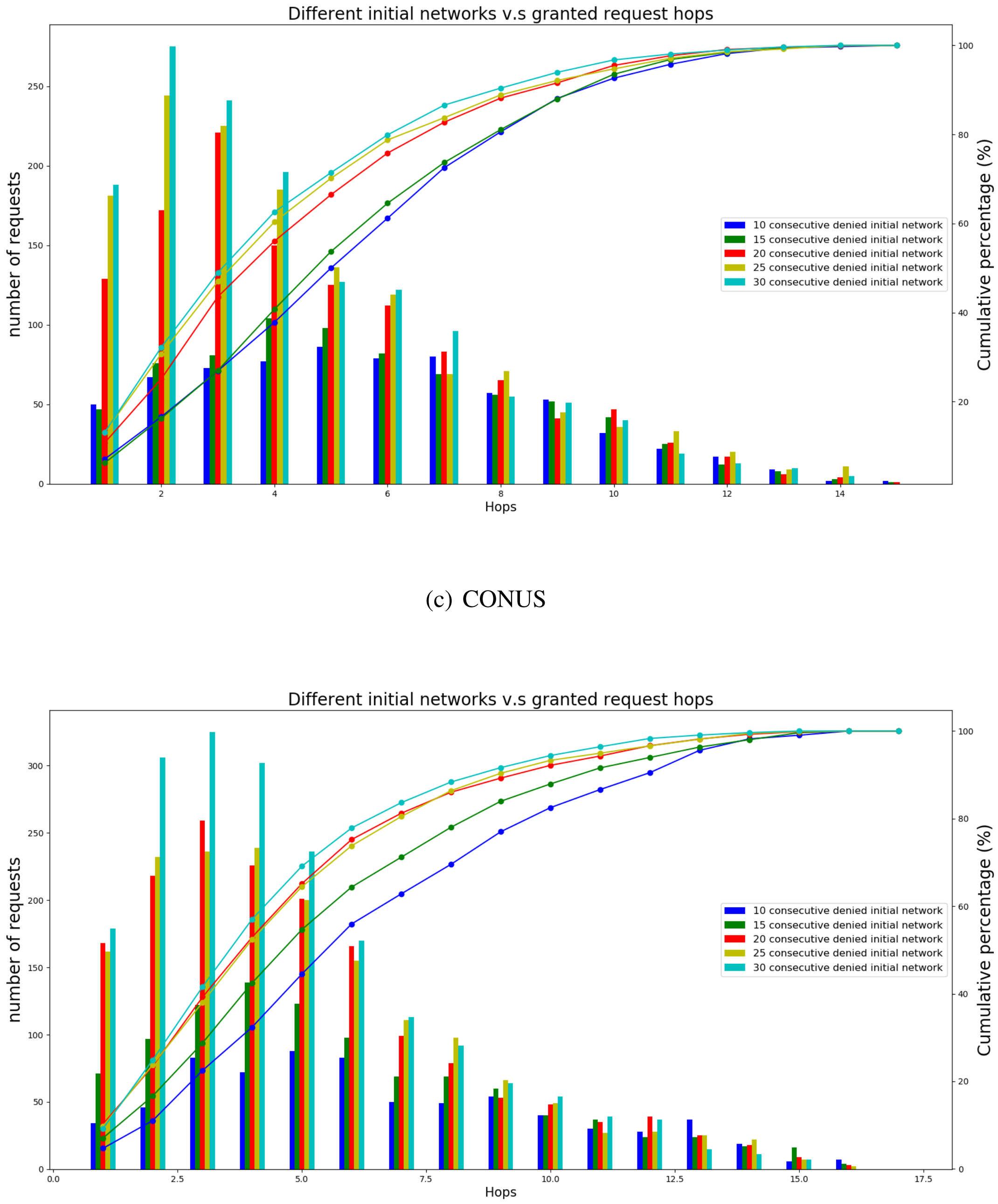
Figure 12: Overall throughput performance under different initial network status



(a) USA



(b) Germany



(c) CONUS

(d) NTT

Figure 13: Path hop bias under different stopping conditions

6.3 Overview of the Various Defragmentation Strategies

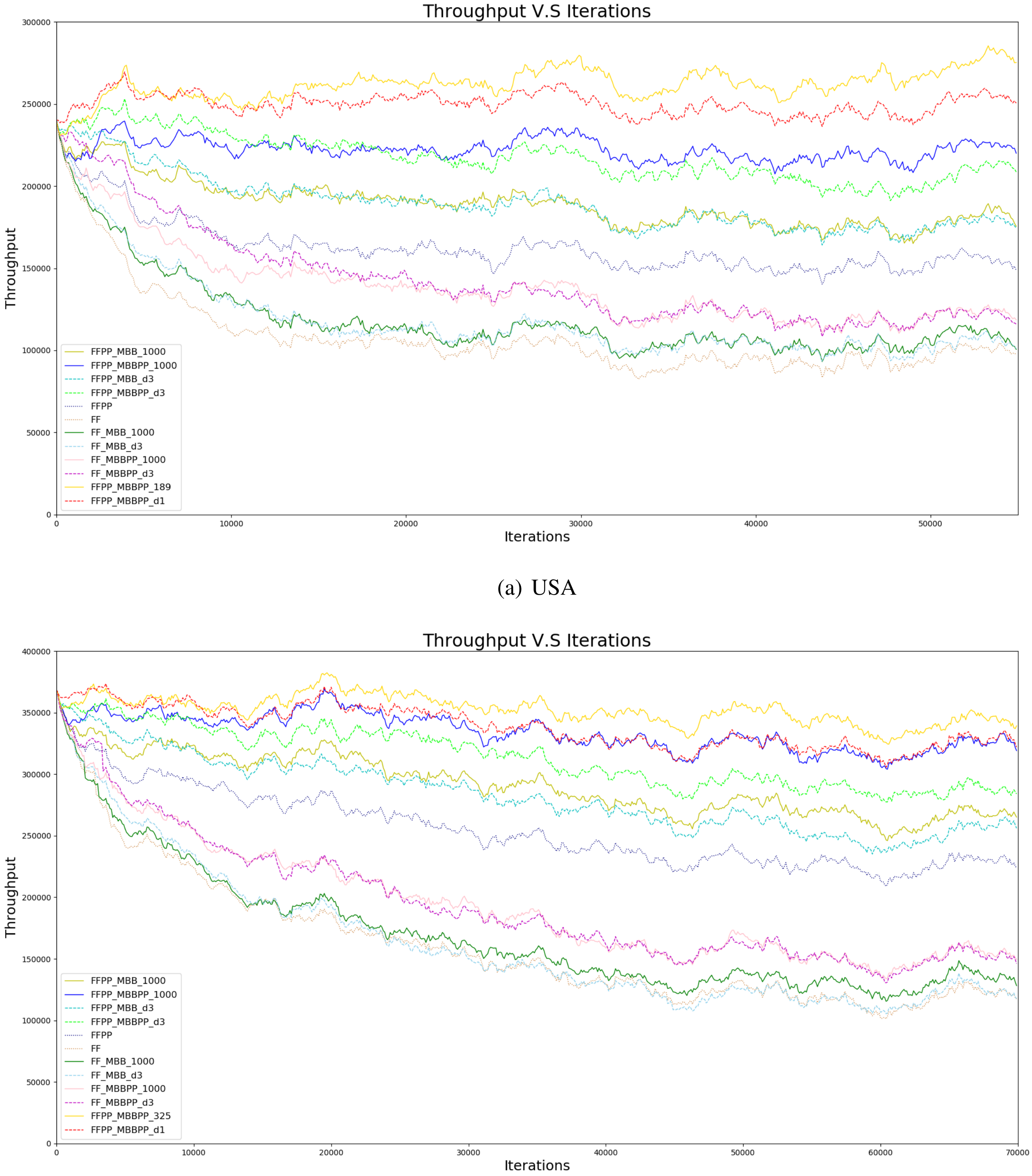
In the previous section, we found that the initial network does not have a significant impact on the performance of RMSA provisioning and defragmentation strategy. Therefore, we select the initial network with stopping condition i=10, since it has a lower bias on longer hop requests and converges faster.

We compare all the RMSA provisioning strategies, i.e., FF, FFPP, FF_MBB, FF_MBBPP, FFPP_MBB, FFPP_MBBPP. Recalling that reactive push-pull defragmentation is triggered when a new incoming request is denied, in this thesis, we design 2 proactive defragmentation triggering policies, i.e., time-driven and throughput-driven. We define that time-driven proactive defragmentation is triggered when the optical network reaches the predefined number of adding and dropping sequences. The throughput-driven defragmentation policy, on the other hand, is triggered when the overall throughput reaches the predefined percentage of decreases.

In this section, time-driven proactive defragmentation is triggered every 1000 request adding and dropping sequence (denoted with suffix 1000) and throughput-driven proactive defragmentation is triggered when the overall throughput has a 3% decrease (denoted with suffix $d3$). In addition, in order to 'approximate the best performance of the proposed combinations of RMSA provisioning and defragmentation strategies', we use FFPP_MBBPP strategy so that proactive defragmentation can be triggered when the network throughput decreases by 1% (denoted with suffix $d1$) in throughout-driven triggering policies. Moreover, in time-driven policies, we set the frequency of defragmentation to be similar to the throughput-driven one, e.g., in Figure 14(a), we have FFPP_MBBPP_189, where the suffix 189 means that proactive defragmentation is triggered every 189 adding and dropping sequences, and the number of the proactive defragmentation events are similar to the frequency of throughput-driven policy.

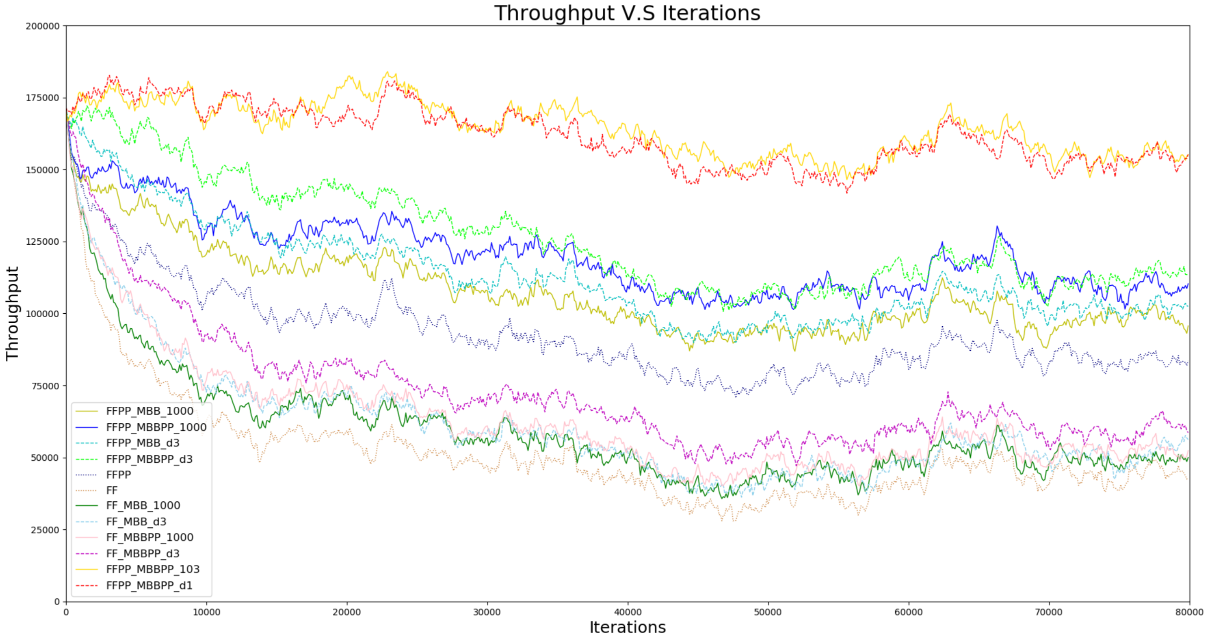
Simulation results are shown in Figure 14. The X-axis represents the time units, which each have an average of 1 request adding and 1 request dropping, and the Y-axis shows the network overall throughput. As we can observe from Figure 14, both reactive and proactive push-pull defragmentation have impacts on network overall throughput. Specifically, RMSA provisioning strategies with reactive push-pull defragmentation have significant improvements on the network throughput. In addition, in every plot in Figure 14, the RMSA strategies without proactive push-pull defragmentation are FFPP and FF, and represent the lower bound for RSMA strategies with and without reactive push-pull respectively. We use these 2 RMSA strategies to investigate the impact of the reactive push-pull defragmentation, the details of which are discussed in Section 6.4.

For proactive push-pull defragmentation, it is obvious to see that RMSA strategies with proactive defragmentation have better performance compared with those without proactive defragmentation, e.g., FFPP_MBB and FFPP, etc. Furthermore, under the same proactive defragmentation triggering strategy, proactive defragmentation with push-pull have better performance compared with those do not have push-pull, for example, FFPP_MBB_d3 and FFPP_MBBPP_d3. However, different proactive defragmentation triggering strategies have different performances. In Section 6.5, we investigate 2 proactive defragmentation triggering strategies under similar push-pull triggering events.

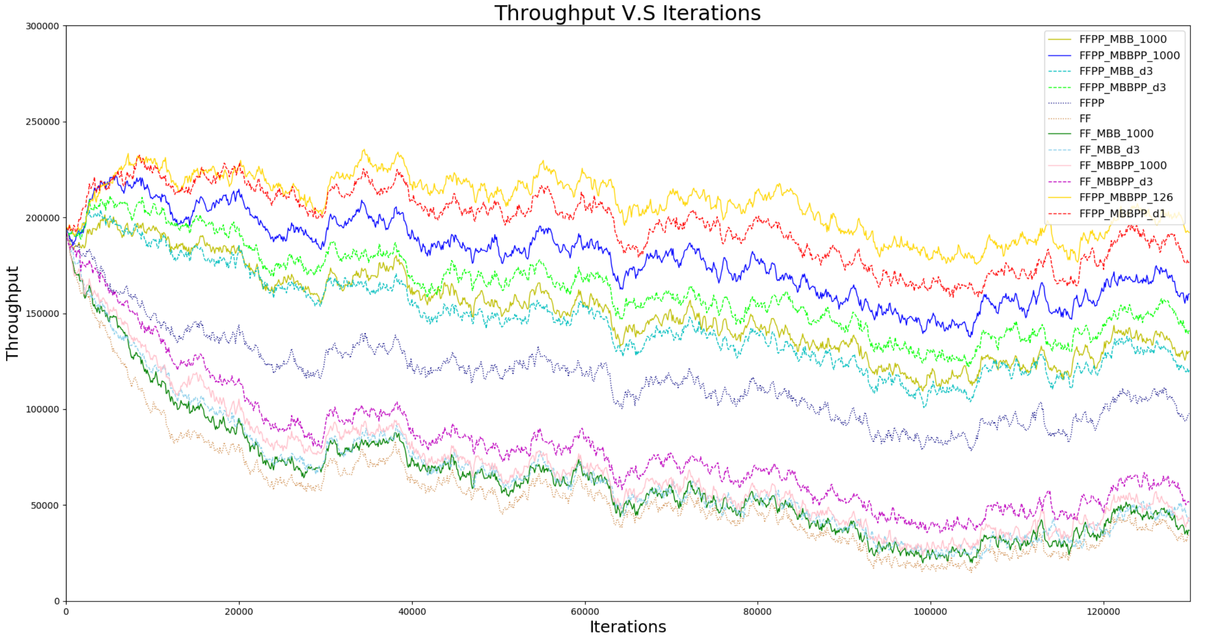


(a) USA

(b) Germany



(c) CONUS



(d) NTT

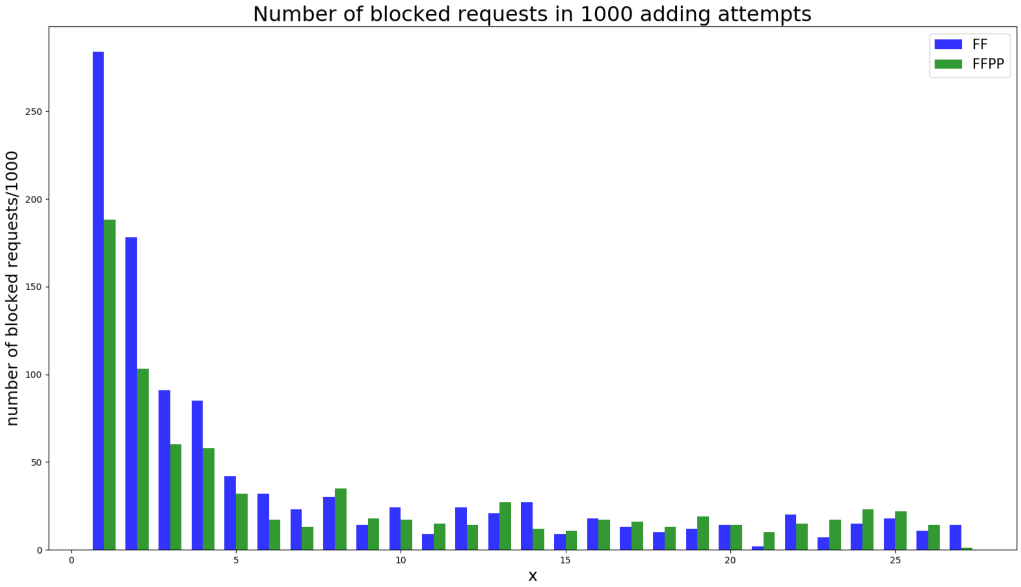
Figure 14: Comparison of all RMSA planning and defragmentation strategies

6.4 Impact of Reactive Push-Pull Defragmentation

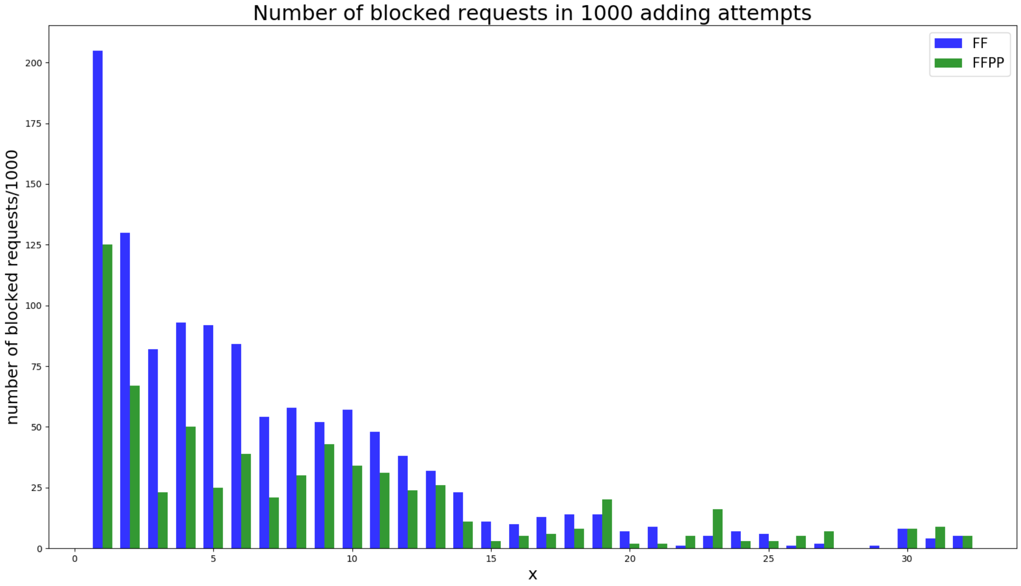
In this section, we comment on the results of reactive push-pull defragmentation in depth. As we can observe from Figure 14, RMSA strategies with reactive push-pull defragmentation have at least 1.5 times overall throughput compared with those without reactive push-pull defragmentation. Therefore, reactive push-pull defragmentation has significant improvements on network throughput.

Figure 15 illustrates the blocking ratio of FF and FFPP in every 1000 request adding events. The X-axis represents n^{th} 1000 adding events and the Y-axis shows the number of the blocked requests. The blue and green bar represents the number of blocked requests in every 1000 adding events of FF and FFPP respectively. As we can observe from Figure 15, the measured values are different on different network topologies. However, as time goes by, the number of denied requests for both FF and FFPP decreases since the overall throughput in each topology also decreases, making the network less loaded. Reactive push-pull defragmentation reduces the number of denied requests, especially when the network is heavily loaded.

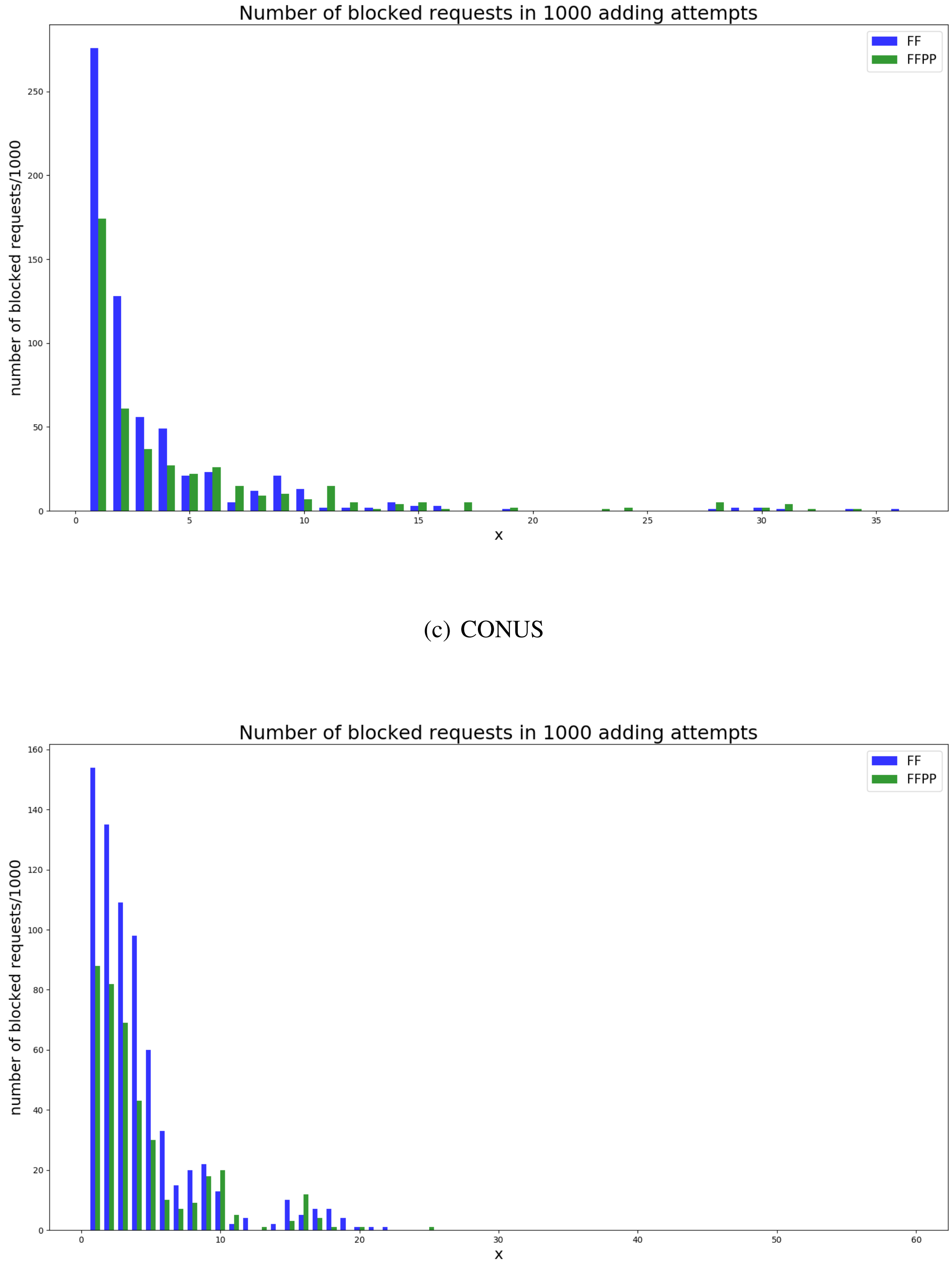
According to the experiment results, we find that push-pull in reactive defragmentation has a quite good performance in terms of improving network throughput and reducing the number of blocked requests.



(a) USA



(b) Germany



(c) CONUS

(d) NTT

Figure 15: Blocking ratio comparison between FF and FFPP

6.5 Impact of Proactive Push-Pull Defragmentation

In this section, we will discuss the impact of proactive push-pull defragmentation. Since proactive defragmentation is performed periodically and is independent on request connection events, a proper triggering policy is essential and necessary to be investigated [23].

We designed two different proactive defragmentation triggering strategies, i.e., time-driven and throughput-driven, and for each strategy, we tested several triggering thresholds. For the time-driven strategy, we tried to do proactive defragmentation every 500, 1000, 1,500 to 2,000 request adding and dropping sequence respectively. For the throughput-driven strategy, proactive defragmentation is performed if the network throughput decreased by 3%, 4%, and 5% respectively.

6.5.1 Proactive Push-Pull with Different Triggering Strategies

We compare proactive defragmentation with different triggering strategies. The results showing in each table cell are represented by a tuple. The 1st element in the tuple represents the number of proactive defragmentations triggered based on the predefined triggering policy. In order to measure the impact of the defragmentation, we capture the spectrum usage improvements, denoted as $\triangle SU$, which equal to the difference between SU before and SU after defragmentation, that is, $SU = \sum\limits_{r \in R} \sum\limits_{\ell \in p_r} w_\ell \, d_r$. The 2nd value in the tuple represents $\triangle SU$.

In addition, the 1st column in each table represents different proactive triggering strategies, for example, 500 means that the proactive defragmentation is triggered every 500 request adding and dropping sequences, d3 means that proactive defragmentation is triggered when the network throughput decreased by 3%, and so on and so forth. Every throughput-driven strategy d1 is associated with a time-driven strategy. The corresponding time-driven strategy has a similar number of proactive defragmentation events and is shown in the 2nd line in each table, e.g., 189 in USA topology and 325 in Germany topology, etc. Both triggering strategies are used to approximate the upper bound of the added value of proactive defragmentation. Results are recorded from Table 3 to Table 6.

	FF_MBB	FF_MBBPP	FFPP_MBB	FFPP_MBBPP
189				(292, 132609.07)
500	(110, 141012.47)	(110, 314463.27)	(110, 111314.66)	(110, 314571.80)
1000	(55, 224954.74)	(55, 502862.28)	(55, 191014.40)	(55, 540505.00)
1500	(37, 269967.69)	(37, 603503.07)	(37, 242515.12)	(37, 686919.10)
2000	(28, 300600.17)	(28, 641728.10)	(28, 281157.75)	(28, 765879.48)
d1				(290, 128079.85)
d3	(176, 92499.03)	(130, 254403.61)	(68, 157717.94)	(54, 530051.55)
d4	(105, 137174.29)	(81, 367482.93)	(44, 207546.70)	(39, 626016.31)
d5	(70, 181872.33)	(60, 444639.29)	(29, 257910.16)	(28, 696354.50)

Table 3: Added value of proactive push-pull in the USA topology

	FF_MBB	FF_MBBPP	FFPP_MBB	FFPP_MBBPP
325				(219, 4380.39)
500	(142, 2278.43)	(142, 5729.31)	(142, 2469.25)	(142, 6472.09)
1000	(71, 3854.41)	(71, 9862.87)	(71, 4468.16)	(71, 11787.21)
1500	(46, 4949.43)	(46, 12474.84)	(46, 5970.27)	(46, 16062.18)
2000	(35, 5644.37)	(35, 14588.54)	(35 7211.52)	(35, 19468.75)
d1				(220, 4171.55)
d3	(154, 1957.41)	(99, 7063.46)	(47, 5401.83)	(38, 17489.28)
d4	(80, 3084.68)	(62, 9854.52)	(27, 7393.33)	(23, 21865.45)
d5	(62, 3613.94)	(42, 11635.40)	(20, 8780.23)	(18, 24045.02)

Table 4: Added value of proactive push-pull in the Germany topology

	FF_MBB	FF_MBBPP	FFPP_MBB	FFPP_MBBPP
103				(776, 59954.78)
500	(159, 105505.28)	(159, 220904.67)	(159, 105162.95)	(159, 250784.46)
1000	(79, 143368.14)	(79, 291228.07)	(79, 159850.77)	(79, 392253.57)
1500	(52, 149454.27)	(52, 291194.12)	(52, 189157.07)	(52, 452778.96)
2000	(39, 151339.35)	(39, 308015.27)	(39, 201288.54)	(39, 472598.58)
d1				(775, 59284.17)
d3	(747, 30667.92)	(528, 83318.23)	(188, 86605.41)	(161, 227882.56)
d4	(429, 47568.62)	(291, 131945.83)	(125, 115938.81)	(103, 300305.81)
d5	(294, 62700.74)	(233, 150115.87)	(77, 145322.91)	(76, 357070.62)

Table 5: Added value of proactive push-pull in the CONUS topology

	FF_MBB	FF_MBBPP	FFPP_MBB	FFPP_MBBPP
126				(1100, 1140.98)
500	(260, 1007.77)	(260, 2886.44)	(260, 1618.01)	(260, 4024.25)
1000	(130, 1526.50)	(130, 3898.19)	(130, 2654.06)	(130, 6729.80)
1500	(87, 1488.14)	(87, 4015.28)	(87, 3281.79)	(87, 8365.99)
2000	(65, 1482.28)	(65, 4013.03)	(65, 3680.36)	(65, 9356.70)
d1				(1108, 1086.13)
d3	(1873, 176.64)	(1062, 899.13)	(259, 1508.79)	(189, 4676.17)
d4	(1317, 211.79)	(765, 1128.93)	(156, 2083.66)	(128, 5796.40)
d5	(794, 440.11)	(506, 1451.87)	(107, 2600.52)	(95, 6287.08)

Table 6: Added value of proactive push-pull in the NTT topology

According to the results from the tables above. We can summarize as follows: In throughput driven strategies, proactive defragmentation with push-pull always have less triggering events (i.e, FF_MBB v.s FF_MBBPP or FFPP_MBB v.s FFPP_MBBPP). It is easy to understand since proactive defragmentation with push-pull has higher average spectrum improvements and the network work has more free spectrum allocations to grant upcoming requests. Moreover, if we decrease the number of proactive defragmentation events, we always get a higher $\triangle SU$, which means that proactive defragmentation does have improvements in spectrum usage. However, $\triangle SU$ does not keep increasing with the number of triggering events decreasing, e.g., FF_MBB (1500) and FF_MBB (2000). Most importantly, if we compare time-driven methods with throughput-driven methods under a similar number of proactive events, for example, FFPP_MBBPP (1000) v.s FFPP_MBBPP (d3) in USA topology, FFPP_MBB (1500) v.s FFPP_MBB (d3) in Germany topology, FFPP_MBBPP (500) v.s FFPP_MBBPP (d3) in CONUS topology and FFPP_MBBPP (1000) v.s FFPP_MBBPP (d4) in NTT topology, etc, we can see that time-driven strategies always have a better performance in terms of spectrum usage improvements, which can be explained by the fact that throughput-driven strategies trigger some 'unnecessary' or 'inefficient' defragmentation events, which we will discuss in the following subsection.

6.5.2 Time Driven vs. Throughput Driven Triggering

In the previous section, we find that the time-driven strategy always has a better $\triangle SU$ compared with throughput-driven strategy under a similar number of proactive defragmentation triggering events. In this section, we will discuss the overall throughput performance of two defragmentation triggering strategies. Figure 16 shows the comparison between the time-driven proactive defragmentation strategy and the throughput-driven proactive defragmentation strategy. For each network topology, we choose 3 pairs of RMSA strategies to compare.

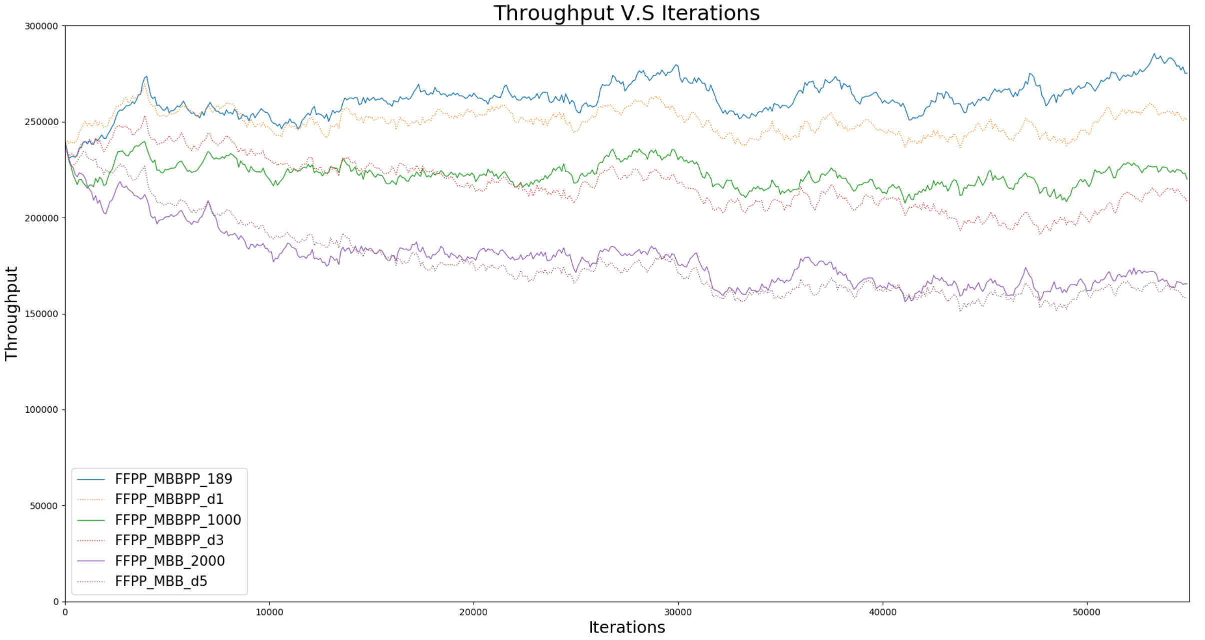
In the USA topology, we compare FFPP_MBBPP (189, with 292 proactive defragmentations) with FFPP_MBBPP ($d1$, with 290 proactive defragmentations), FFPP_MBBPP (1000, with 55 proactive defragmentations) with FFPP_MBBPP ($d3$, with 54 proactive defragmentations) and FFPP_MBB (2000, with 28 proactive defragmentations) with FFPP_MBB ($d5$, with 29 proactive defragmentations). Results are shown in Figure 16(a).

In the Germany topology, we compare FFPP_MBBPP (325, with 219 proactive defragmentations) with FFPP_MBBPP ($d1$, with 220 proactive defragmentations), FFPP_MBBPP (2000, with 35 proactive defragmentations) with FFPP_MBBPP ($d3$, with 38 proactive defragmentations) with FFPP_MBB (1500, with 46 proactive defragmentations) with FFPP_MBB ($d3$, with 47 proactive defragmentations). Results are shown in Figure 16(b).

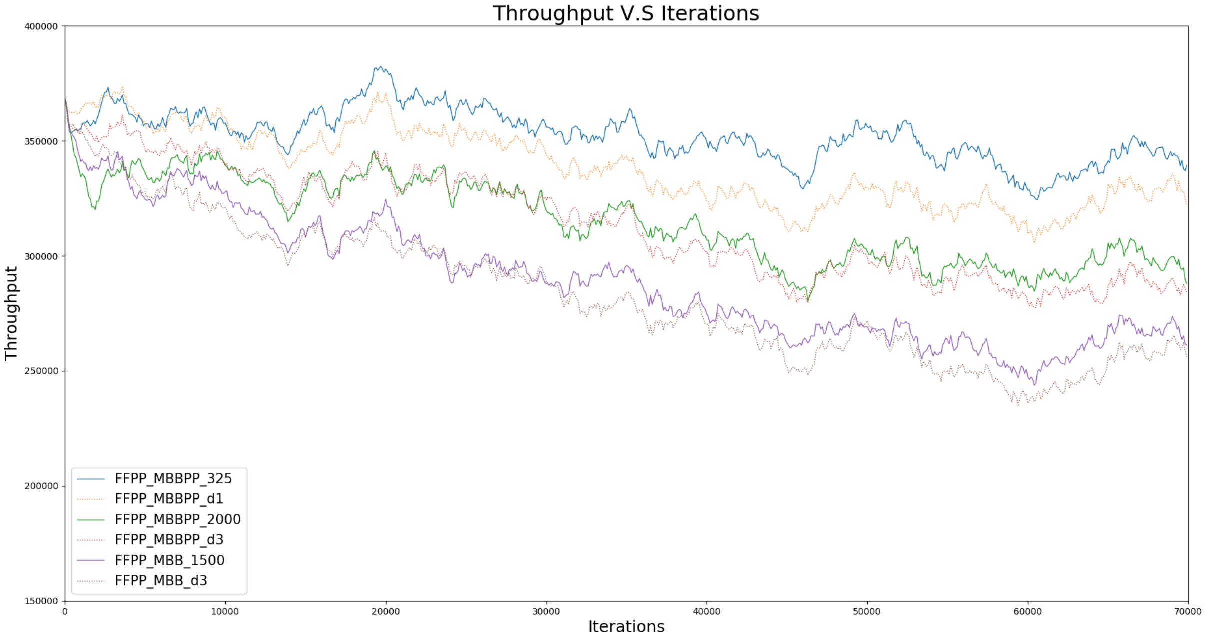
For the CONUS topology, we compare FFPP_MBBPP (103, with 776 proactive defragmentations) with FFPP_MBBPP ($d1$, with 775 proactive defragmentations), FFPP_MBBPP (500, with 159 proactive defragmentations) with FFPP_MBBPP ($d3$, with 161 proactive defragmentations) and FFPP_MBB (1000, with 79 proactive defragmentations) with FFPP_MBB ($d5$, with 77 proactive defragmentations). Results are shown in Figure 16(c).

In terms of the NTT topology, we compare FFPP_MBBPP (126, with 1100 proactive defragmentations) with FFPP_MBBPP ($d1$, with 1108 proactive defragmentations), FFPP_MBBPP (1000, with 130 proactive defragmentations) with FFPP_MBBPP ($d4$, with 128 proactive defragmentations) and FFPP_MBB (500, with 260 proactive defragmentations) with FFPP_MBB ($d3$, with 259 proactive defragmentations). Results are shown in Figure 16(d).

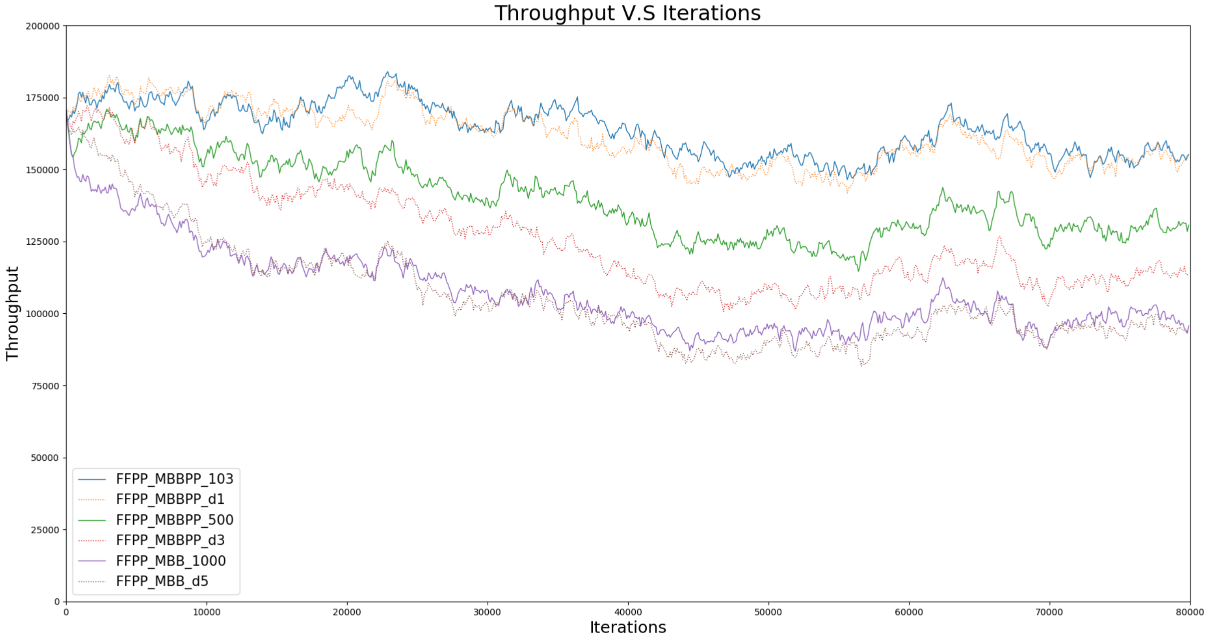
As we can observe from Figure 16, throughput-driven method always have better defragmentation performance when the provisioning network is heavily loaded. It is easy to understand since throughput-driven strategies are based on the network status. In the beginning, the network is heavily loaded and hard to grant new requests, therefore, throughput-driven strategy triggers more defragmentations compared with the time-driven strategy. However, as time goes by, we can observe that the time-driven method always has better throughput performance. This can be explained by the fact that the throughput-driven method triggers some inefficient defragmentations when the network status becomes stable.



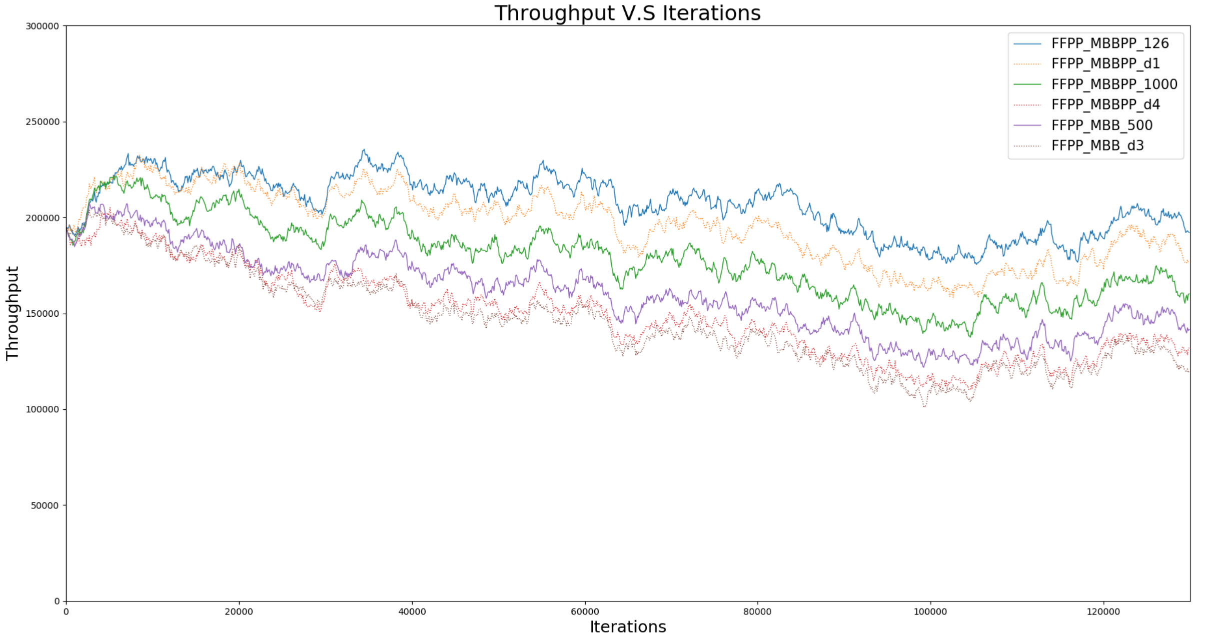
(a) USA



(b) Germany



(c) CONUS



(d) NTT

Figure 16: Comparison of network throughput performance between time-driven and throughput-driven strategies

6.6 Push-Pull Delay

In the previous sections, we discussed the added values of reactive and proactive push-pull defragmentation. However, delays are associated with push-pull. In this section, we will discuss defragmentation delays in reactive and proactive defragmentations.

6.6.1 Reactive Push-Pull Delay

In this section, we capture the delay of the reactive push-pull defragmentation. Results are recorded from Table 7 to Table 10. We record the results into triples: the 1^{st} and 2^{nd} elements are the minimum and the maximum delay of push-pull during the request adding and dropping sequence and the 3^{rd} element represents the average delay. The minimum delay for all the topology is 1 shifting distance, whereas the maximum delays are various.

For the USA topology, the reactive push-pulll delays are shown in Table 7, the maximum delays are between 16 and 26, this is because of the maximum request demands in USA topology is 32 frequency slots (request with 400 Gbps using a routing path which distance longer than 4000 km, Table 2), therefore, push-pull needs more shifting distance in order to grant those requests, and average delays in USA topology are between 7.32 and 7.66. Reactive push-pull delays of Germany topology are shown in Table 8, the maximum delays are between 4 and 7, and the average delays are between 2.42 and 2.71. Delays in Germany are less than in the USA since requests in Germany always use shorter routing paths associated with higher level modulation formats. Results of CONUS topology are shown in Table 9, which are similar to the results of USA, the maximum delays vary from 16 to 18 shifting distances and average delays vary from 8.20 to 8.37. For the NTT topology, because of the modulation, request demands in NTT are not as large as the other topologies: the maximum shifting distance is 4, and the average delays vary from 1.52 to 1.80. We can summarize the reactive push-pull delay as follows:

The RMSA strategies with proactive defragmentation always have larger delay values, in addition, proactive defragmentations with push-pull tend to have larger delays since RMSA strategies with proactive push-pull defragmentation always have larger network throughput and more provisioned requests. Moreover, reactive push-pull delay under same RMSA strategy depends on proactive defragmentation frequency, RMSA strategies with more proactive defragmentations tend to have larger delays. Furthermore, the proactive triggering strategy does not have a significant impact on average delays under the same RMSA strategy. Reactive push-pull delays vary on different network topologies, a network with longer paths tends to have larger delays since a long routing path may use lower-level modulation and higher slot requirements.

	FFPP	FFPP_MBB	FFPP_MBBPP
	(min delay, max delay, average delay)		
189	(1, 16, 7.32)		(1, 20, 7.66)
500		(1, 17, 7.41)	(1, 23, 7.60)
1000		(1, 14, 7.39)	(1, 19, 7.60)
1500		(1, 22, 7.38)	(1, 25, 7.53)
2000		(1, 19, 7.37)	(1, 23, 7.50)
d1			(1, 25, 7.65)
d3		(1,20, 7.38)	(1, 26, 7.57)
d4		(1, 18, 7.38)	(1, 18, 7.53)
d5		(1, 16, 7.36)	(1, 16, 7.48)

Table 7: Reactive delay in USA

	FFPP	FFPP_MBB	FFPP_MBBPP
	(min delay, max delay, average delay)		
325	(1, 6, 2.42)		(1, 5, 2.71)
500		(1, 4, 2.52)	(1, 6, 2.69)
1000		(1,7, 2.50)	(1,7, 2.67)
1500		(1, 6, 2.49)	(1, 4, 2.67)
2000		(1, 4, 2.47)	(1, 4, 2.66)
d1			(1, 5, 2.68)
d3		(1, 4, 2.47)	(1, 5, 2.65)
d4		(1, 4, 2.45)	(1, 4, 2.63)
d5		(1, 4, 2.45)	(1, 4, 2.63)

Table 8: Reactive delay in Germany

	FFPP	FFPP_MBB	FFPP_MBBPP
	(min delay, max delay, average delay)		
103	(1, 16, 8.20)		(1, 18, 8.37)
500		(1, 16, 8.27)	(1, 17, 8.36)
1000		(1, 16, 8.26)	(1, 16, 8.35)
1500		(1, 16, 8.26)	(1, 16, 8.33)
2000		(1, 15, 8.26)	(1, 15, 8.33)
d1			(1, 18, 8.36)
d3		(1, 16, 8.26)	(1, 16, 8.34)
d4		(1, 16, 8.25)	(1, 16, 8.33)
d5		(1, 16, 8.24)	(1, 16, 8.33)

Table 9: Reactive delay in CONUS

	FFPP	FFPP_MBB	FFPP_MBBPP
	(min delay, max delay, average delay)		
126	(1, 3, 1.52)		(1, 4, 1.80)
500		(1, 3, 1.64)	(1, 4, 1.78)
1000		(1, 3, 1.63)	(1, 3, 1.77)
1500		(1, 3, 1.59)	(1, 3, 1.75)
2000		(1, 3, 1.58)	(1, 3, 1.75)
d1			(1, 4, 1.79)
d3		(1, 3, 1.57)	(1, 4, 1.75)
d4		(1, 3, 1.55)	(1, 4, 1.73)
d5		(1, 3, 1.55)	(1, 3, 1.70)

Table 10: Reactive delay in NTT

6.6.2 Proactive Push-Pull Delay

In this section, we discuss the proactive delay of push-pull. Results are shown from Table 11 to Table 14. Results are recorded into triples: the 1st element represents the minimum sum delays, maximum sum delays are shown by the 2nd element and the average sum delays are shown in the 3rd positions. The sum delay is defined as

$$\Delta_j = \sum_i^n \delta_i^j$$

where n is the number of triggered push-pull defragmeantion in jth proactive defragmentation.

	FF_MBBPP	FFPP_MBBPP
	(min sum delay, max sum delay, Avg sum delay)	
189		(42, 578, 263.23)
500	(25, 288, 115.47)	(168, 668, 394.53)
1000	(32, 345, 135.78)	(162, 592. 409.41)
1500	(20, 212, 119.58)	(255, 587. 408.27)
2000	(18, 292, 111.33)	(254, 534, 396.00)
d1		(16, 579, 282.67)
d3	(2, 338, 79.53)	(88, 589, 356.00)
d4	(8, 367, 109.34)	(133, 607, 341.69)
d5	(10, 399, 106.69)	(88, 490, 337.18)

Table 11: Proactive delay in USA

	FF_MBBPP	FFPP_MBBPP
	(min sum delay, max sum delay, Avg sum delay)	
189		(2, 279, 44.13)
500	(2, 177, 32.56)	(3, 276, 58.08)
1000	(1, 216, 38.55)	(12, 357, 108.75)
1500	(2, 272, 48.00)	(22, 352, 123.25)
2000	(4, 260, 48.91)	(18, 343, 150.35)
d1		(2, 306, 46.06)
d3	(3, 286, 44.60)	(17, 307, 156.25)
d4	(4, 239, 59.00)	(3, 313, 196.00)
d5	(3, 256, 56.00)	(16, 318, 198.50)

Table 12: Proactive delay in Germany

	FF_MBBPP	FFPP_MBBPP
	(min sum delay, max sum delay, Avg sum delay)	
103		(1, 295, 112.21)
500	(1, 222, 76.82)	(5, 303, 190.08)
1000	(3, 203, 82.10)	(5, 293, 250.94)
1500	(2, 239, 94.43)	(17, 409, 258.81)
2000	(13, 211, 114.40)	(28, 461, 286.64)
d1		(1, 292, 94.61)
d3	(1, 215, 34.23)	(7, 369. 168.00)
d4	(1, 251, 36.36)	(1, 404, 182.10)
d5	(3, 228, 49.76)	(1, 452, 219.91)

Table 13: Proactive delay in CONUS

	FF_MBBPP	FFPP_MBBPP
	(min sum delay, max sum delay, Avg sum delay)	
126		(1, 103, 24.34)
500	(1, 99, 24.26)	(1, 166, 38,86)
1000	(7, 57, 30.86)	(1, 153, 46.88)
1500	(7, 75, 33.8)	(2, 184, 52.53)
2000	(15, 109, 42.71)	(2, 129, 50.96)
d1		(1, 185, 25.96)
d3	(2, 86, 31.06)	(1, 135, 39.65)
d4	(3, 53, 31.30)	(7, 128, 46.91)
d5	(5, 105, 38.10)	(1, 123, 44.48)

Table 14: Proactive delay in NTT

As we observe from the tables above, FFPP_MBBPP has larger delays compared with FF_MBBPP under the same proactive defragmentation frequency. It is easy to understand since FFPP_MBBPP always has a larger set of provisioned requests and the proactive defragmentation process involves more requests in order to reconfigure a request by using push-pull. Therefore, the minimum, maximum and average sum delays are relatively larger. In addition, just like the delays of reactive push-pull, proactive push-pull delays vary on different network topologies; a network with longer paths tends to have larger sum delays because of the modulation formats.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we have studied online RMSA with push-pull in both proactive and reactive defragmentation under several RMSA request provisioning and defragmentation strategies.

Simulation results show that the RMSA provisioning strategy with reactive push-pull defragmentation has a significant improvement in terms of network throughput and request blocking ratio. On the other hand, proactive defragmentation has shown significant improvements in terms of spectrum usage. Moreover, proactive defragmentation with push-pull always has fewer triggering events and better network throughput performances compared with the strategies without push-pull. If the networking is not heavily provisioned, only make-before-break proactive defragmentation is a better option since it will not introduce defragmentation delays. Therefore, reactive push-pull defragmentation can be used to improve network throughput whereas proactive defragmentation can be used to maintain network throughput at an acceptable level. In addition, throughput-driven proactive defragmentation has a better performance in terms of network throughput when the network is heavily loaded, whereas time-driven proactive defragmentation is a better option when the network status becomes stable. We also captured delays in reactive and proactive push-pull, and results show that delays in the same network topology are similar but vary in different network topologies. Therefore, delays in both proactive and reactive push-pull depend on the modulation and network characteristics. Furthermore, in the same network topology, both reactive and proactive push-pull delays depend on the network throughput and the number of granted requests.

7.2 Contributions

- We evaluated reactive and proactive push-pull defragmentation in an online RMSA environment.
- We investigated different combinations of RMSA request provisioning and defragmentation strategies. A better combination option depends on the network topology characteristics and network status.
- We proposed a cheap and easy proactive push-pull defragmentation with rerouting scenario.
- The simulation is based on a large directed and weighted network topology and modulations are also taken into account

7.3 Future Work

Future research directions for the problem of online RMSA with push-pull might focus on the three following axes.

- Optimizing another criterion instead of the delay in push-pull spectrum location selection. This criterion might be the total number of requests shifted in order to empty space for the new arriving request or the number of the granted requests involved in this defragmentation process.
- Considering investigating the other parameters for triggering proactive defragmentation, e.g., blocking ratio, number of granted requests, etc.

- Working on designing an ILP model under current online RMSA provisioning and defragmentation environment and comparing the current results with the optimal solutions.

References

[1] R. Almeida, R. Delgado, C. J. Bastos-Filho, D. Chaves, H. A. Pereira, and J. Martins-Filho. An evolutionary spectrum assignment algorithm for elastic optical networks. In *ICTON*, pages 1–3. IEEE, 2013.

[2] B. Chatterjee, N. Sarma, and E. Oki. Routing and spectrum allocation in elastic optical networks: a tutorial. *IEEE Communications Surveys & Tutorials*, 17(3):1776–1800, 2015.

[3] B. C. Chatterjee, N. Sarma, and E. Oki. Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 17(3):1776–1800, 2015.

[4] K. Christodoulopoulos, I. Tomkos, and E. A. Varvarigos. Routing and spectrum allocation in ofdm-based optical networks with elastic bandwidth allocation. In *GLOBECOM*, pages 1–6, 2010.

[5] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos. Elastic bandwidth allocation in flexible OFDM-based optical networks. *Journal of Lightwave Technology*, 29(9):1354 – 1366, May 2011.

[6] D. Coudert, B. Jaumard, and F. Moataz. Dynamic routing and spectrum assignment with non-disruptive defragmentation. In *16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, pages 1–4, 2014.

[7] F. Cugini, M. Secondini, N. Sambo, G. Bottari, G. Bruno, P. Iovanna, and P. Castoldi. Push-pull technique for defragmentation in flexible optical networks. In *NFOEC*, pages 1–3, 2012.

[8] F. Cugini, F. Paolucci, G. Meloni, G. Berrettini, M. Secondini, F. Fresi, N. Sambo, L. Poti, and P. Castoldi. Push-pull defragmentation without traffic disruption in flexible grid optical networks. *Journal of Lightwave Technology*, 31(1):125–133, 2013.

[9] A. Eira, J. Pedro, D. Fonseca, F. J. Arribas, J. Fernandez-Palacios, I. L. Polo, D. Schmuhl, S. Spaelter, D. Marzo, and M. Bohn. Defragmentation of fixed/flexible grid optical networks. In *FNMS*, pages 1–10, 2013.

[10] J. Enoch and B. Jaumard. Towards optimal and scalable solution for routing and spectrum allocation. In *International Network Optimization Conference - INOC*, pages 1–8, 2017.

[11] J. Enoch and B. Jaumard. Towards optimal and scalable solution for routing and spectrum allocation. *Electronic Notes in Discrete Mathematics (ENDM)*, 64C:335–344, 2018.

[12] S. Fernández-Martínez, B. Barán, and D. P. Pinto-Roa. Spectrum defragmentation algorithms in elastic optical networks. *Optical Switching and Networking*, 34:10–22, 2019.

[13] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[14] D. Hunter and D. Marcenac. Dynamic routing, rearrangement, and defragmentation in wdm ring networks. In *Optical Fiber Communication Conference*, pages 168–170, 2000.

[15] M. Jinno, B. Kozicki, H. Takara, A. Watanabe, Y. Sone, T. Tanaka, and A. Hirano. Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network. *IEEE Communications Magazine*, 48(8):138–145, 2010.

[16] M. Klinkowski and K. Walkowiak. Routing and spectrum assignment in spectrum sliced elastic optical path network. *Communications Letters*, 15(8):884–886, 2011.

[17] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[18] S. F. Martínez and D. P. Pinto-Roa. Performance evaluation of non-hitless spectrum defragmentation algorithms in elastic optical networks. In *CLEI*, pages 1–8. IEEE, 2017.

[19] D. Moniz, A. Eira, A. de Sousa, and J. Pires. On the comparative efficiency of non-disruptive defragmentation techniques in flexible-grid optical networks. *Optical Switching and Networking*, 25:149–159, 2017.

[20] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization*, 2007.

[21] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference*, 2007.

[22] A. N. Patel, P. N. Ji, J. P. Jue, and T. Wang. Survivable transparent flexible optical WDM (FWDM) networks. In *OFC*, pages 1–3. Optical Society of America, 2011.

[23] M. Presi, M. Rannello, M. Artiglia, I. Tomkos, I. Cano, J. Prat, and E. Ciaramella. Hitless wavelength assignment in filterless optical access networks. In *ICTON*, pages 1–4, 2016.

[24] R. Proietti, C. Qin, B. Guan, Y. Yin, R. P. Scott, R. Yu, and S. Yoo. Rapid and complete hitless defragmentation method using a coherent RX LO with fast wavelength tracking in elastic optical networks. *Optics express*, 20(24):26958–26968, 2012.

[25] T. Takagi, H. Hasegawa, K. Sato, Y. Sone, A. Hirano, and M. Jinno. Disruption minimized spectrum defragmentation in elastic optical path networks that adopt distance adaptive modulation. In *ECOC*, pages 1–2, 2011.

[26] T. Takagi, H. Hasegawa, K.-i. Sato, Y. Sone, B. Kozicki, A. Hirano, and M. Jinno. Dynamic routing and frequency slot assignment for elastic optical path networks that adopt distance adaptive modulation. In *OFC*, pages 1–3. IEEE, 2011.

[27] S. Talebi, F. Alam, I. Katib, M. Khamis, R. Salama, and G. N. Rouskas. Spectrum management techniques for elastic optical networks: A survey. *Optical Switching and Networking*, 13:34–48, 2014.

[28] I. Tomkos, S. Azodolmolky, J. Sole-Pareta, D. Careglio, and E. Palkopoulou. A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges. *Proceedings of the IEEE*, 102(9):1317–1337, 2014.

[29] X. Wan, L. Wang, N. Hua, H. Zhang, and X. Zheng. Dynamic routing and spectrum assignment in flexible optical path networks. In *OSA*, pages 1–3, 2011.

[30] X. Wan, N. Hua, and X. Zheng. Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *Journal of Optical Communications and Networking*, 4(8):603–613, 2012.

[31] R. Wang and B. Mukherjee. Spectrum management in heterogeneous bandwidth networks. In *GLOBECOM*, pages 2907–2911, 2012.

[32] R. Wang and B. Mukherjee. Provisioning in elastic optical networks with non-disruptive defragmentation. In *Conference on Optical Network Design and Modeling - ONDM*, pages 1–6, December 2013.

[33] R. Wang and B. Mukherjee. Provisioning in elastic optical networks with non-disruptive defragmentation. *Journal of Lightwave Technology*, 31(15):2491–2500, 2013.

[34] X. Wang, Q. Zhang, I. Kim, P. Palacharla, and M. Sekiya. Blocking performance in dynamic flexible grid optical networks-what is the ideal spectrum granularity? In *ECOC*, pages 1–2, 2011.

[35] Y. Wang, X. Cao, and Q. Hu. Routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *ICC*, pages 1–5, 2011.

[36] Y. Wang, J. Zhang, Y. Zhao, J. Wang, and W. Gu. Routing and spectrum assignment by means of ant colony optimization in flexible bandwidth networks. In *OFC/NFOEC*, pages 1–3, 2012.

[37] S. Yang and F. Kuipers. Impairment-aware routing in translucent spectrum-sliced elastic optical path networks. In *EuCNOC*, pages 1–6, 2012.

[38] J. Yen. Finding the *k* shortest loopless paths in a network. *Management Science*, 17(11): 712–716, 1971.

[39] Y. Yu, J. Zhang, Y. Zhao, X. Cao, X. Lin, and W. Gu. The first single-link exact model for performance analysis of flexible grid wdm networks. In *OSA*, pages 1–3, 2013.

[40] G. Zhang, M. De Leenheer, A. Morea, and B. Mukherjee. A survey on ofdm-based elastic core optical networking. *IEEE Communications Surveys & Tutorials*, 15(1):65–87, 2013.

[41] M. Zhang, W. Shi, L. Gong, W. Lu, and Z. Zhu. Bandwidth defragmentation in dynamic elastic optical networks with minimum traffic disruptions. In *ICC*, pages 3894–3898, 2013.

[42] M. Zhang, Y. Yin, R. Proietti, Z. Zhu, and S. B. Yoo. Spectrum defragmentation algorithms for elastic optical networks using hitless spectrum retuning techniques. In *OFC*, pages OW3A–4, 2013.

[43] M. Zhang, C. You, H. Jiang, and Z. Zhu. Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic. *Journal of Lightwave Technology*, 32(5):1014–1023, 2014.