ABSTRACT

TALEBI, SAHAR. On Routing and Spectrum Assignment in Elastic Optical Networks. (Under the direction of Dr. George Rouskas and Dr. Rudra Dutta.)

In recent years, OFDM has been the focus of extensive research efforts in optical transmission and networking, initially as a means to overcome physical impairments in optical communications. However, unlike, say, in wireless LANs or xDSL systems where OFDM is deployed as a transmission technology in a single link, in optical networks it is being considered as the technology underlying the novel elastic network paradigm. Hence, we dedicate this work to study and propose different solution methods for elastic optical networks (EONs) topologies.

In Chapter 1, we discuss the necessity of applying EONs along with all the benefits it brings to the optical networks. We also review the set of constraints (i.e. spectrum contiguity and continuity constraints) that must be satisfied during traffic demands allocation to the spectrum. Then, we classify existing spectrum management techniques for EONs, including offline and online routing and spectrum assignment (RSA), distance-adaptive RSA, fragmentation-aware RSA, traffic grooming, survivability, and multi-path RSA problems in Chapter 2.

We then provide a new insight into the spectrum assignment (SA) problem in mesh networks in Chapter 3 and show that the SA problem transforms to the problem of scheduling multiprocessor tasks on dedicated processors. Similarly, we prove that the RSA problem with fixed-alternate routing in general-topology (mesh) networks (and, hence, in rings as well) is a special case of a multiprocessor scheduling problem.

Based on this new perspective, we show in Chapter 4 that the SA problem in chain (linear) networks is NP-hard for four or more links, but is solvable in polynomial time for three links. We also develop new constant-ratio approximation algorithms for the SA problem in chain networks with the fixed number of links. Finally, we introduce a suite of list scheduling algorithms that are computationally efficient and simple to implement, yet produce solutions that, on average, are within 1-5% of the lower bound.

In Chapter 5, we consider bidirectional ring networks and investigate two problems: (1) the SA problem under the assumption that each demand is routed along a single fixed path (e.g., the shortest path), and (2) the general case of the RSA problem whereby a routing decision along the clockwise and counter-clockwise directions must be made jointly with spectrum allocation. Based on insights from multiprocessor scheduling theory, we investigate the complexity of these two problems and develop new constant-ratio approximation algorithms with a ratio that is strictly smaller than the best known ratio to date.

We also show that the distance-adaptive RSA (DA-RSA) problem in mesh networks is a special case of a multiprocessor scheduling problem in Chapter 6. We then develop a suite

of efficient and effective DA-RSA algorithms that build upon list scheduling concepts. The numerical results indicate that as the network size increases beyond a point that depends on the traffic demand distribution, the spectrum overhead associated with using a long path becomes sufficiently high that it is always best to use the shortest path. Overall, the best algorithm is always within 10-20% of the lower bound, indicating that scheduling concepts can be successfully adapted to address network design problems.

In Chapter 7, we examine the complexity of the DA-RSA problem for general (mesh) networks and derive a set of conditions that makes this problem either NP-hard or polynomially solvable. We also develop an efficient and effective algorithm for mesh networks that builds upon list scheduling concepts in which routing and spectrum allocation decisions are made jointly. We then run an extensive simulations for DA-RSA in mesh networks to estimate the required amount spectrum for different size networks. Our work explores the tradeoffs involved in DA-RSA algorithm design, and opens up new research directions in leveraging the vast literature in scheduling theory to address important and practical problems in network design.

Finally, we discuss future research directions for the SA and RSA problems in EONs in Chapter 8.

On Routing and Spectrum Assignment in Elastic Optical Networks

by
Sahar Talebi

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2015

APPROVED BY:

Dr. Yahya Fathi

Dr. Matthias Stallmann

Dr. George Rouskas
Co-chair of Advisory Committee

Dr. Rudra Dutta
Co-chair of Advisory Committee

DEDICATION

To my dear mother,
Safieh Abdolmohammadi,
and my wonderful sisters,
Sara Talebi and Sepideh Talebi,
for their boundless love and unwavering support.

BIOGRAPHY

Sahar Talebi is a PhD candidate in Operations Research with co-major in Computer Science at North Carolina State University. She received her BS degree in Industrial Engineering from University of Tehran in 2006 and her MS degree in Industrial Engineering from Iran University of Science & Technology in 2010 in Tehran, Iran. She has more than three years of work experience in the field of analytics in oil industries.

In spring 2011, she has joined Operations Research program at the North Carolina State University to pursue her PhD degree. She also received master degree in Operations Research from the North Carolina State University in fall 2014. She interned at Cox Automotive in summer 2015, where she worked on developing new models and algorithms for B2B businesses. She is a member of the Institute for Operations Research and the Management Sciences (INFORMS), the Institute of Electrical and Electronics Engineers (IEEE), and NCSU Women in Computer Science (WiCS). Her research focuses on Network Modeling, Optimization, Scheduling, and Algorithm Development. She has given several research presentations in both Operations Research and Computer Science conferences.

ACKNOWLEDGEMENTS

This dissertation is the end of my PhD journey. It would not have been possible to finish this journey without the support and help of many people. I would like to take this opportunity to recognize them.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor George Rouskas, who has been a tremendous mentor during this journey. His guidance, dedication, kindness, and patience have contributed immensely to my development as an independent researcher. I am deeply grateful for having the opportunity to collaborate with him during my time at the North Carolina State University. There is no doubt that I would not be able to complete this work without his support, guidance, and tutelage. I am forever indebted to him for believing in me, being extremely supportive, and teaching me the importance of good research.

I would like to extend my sincere gratitude to Professor Rudra Dutta for letting me be a part of his team, co-advising me along this way, and assisting me in shaping as a student and researcher. His patience and encouragement remarkably helped me during this journey. It is a great honor and privilege for me to have experience of working with an extremely knowledgeable, inspiring, and kind teacher.

I am also grateful to Professor Yahya Fathi for kindly accepting to be in my doctoral committee and for his supervision during my first year as a graduate student at the North Carolina State University. I admire him for his intelligent suggestions on my work, encouragements, and unceasing support. Words cannot express my thankfulness for his invaluable assistance both during performing this research and through my entire graduate studies.

I owe a word of great appreciation to Professor Matthias Stallmann for accepting to serve on my doctoral committee and generously sharing his knowledge with me. My work has undoubtedly benefited from his constructive comments and creative ideas. I am thankful for his assistance and exceptional willingness to help me to work out the details and improve the quality of my work.

I would also like to express my gratitude to Professor Evripidis Bampis and Doctor Giorgio Lucarelli at Pierre and Marie Curie University (University of Paris VI) for their collaboration and creative suggestions on my work. Their guidance helped me immensely come up with great ideas. I am specially thankful to them for answering my questions and enlightening me along this way.

I feel fortunate to have had the opportunity to be a graduate student in the department of Operations Research and the department of Computer Science at the North Carolina State University. I am also thankful for various fundings supported my studies and research over the last few years. Namely, the department of Industrial & Systems Engineering provided funding in the way of teaching assistantship. I would like to thank the National Science Foundation for supporting my work financially under grant CNS-1113191. I am extremely appreciative to the faculty and staff of the department of Industrial & Systems Engineering, the department of Operations Research, and the department of Computer Science.

I am so blessed and grateful to have many true and real friends all over the world. Many thanks go to my wonderful college friends Maliheh Aramoon, Golshid Baharian, Sara Ghorbani, Negar Golchin and my amazing friends Bahar Aghashiri, Arsineh Galestanian, and Maryam Zaghian who have enriched my life and always supported me emotionally during this journey. I further wish to express my appreciation to fellow friends and graduate students at the North Carolina State University who have always supported me and made this journey memorable. Special thanks go to Zahra Zamani, Atefeh Zamani, Shadi Hassani Goodarzi, Maryam Delavarrafiee, Anahita Kamyab, Roya Yousefelahiyeh, Shakiba Enayati, Golbarg Kazemi, Giti Karimi, Pegah Pooya, Armita Mohammadian, Magreth Mushi, Morteza Amouzgar, Arash Rezaei, Amir Bahmani, Mehdi Mashayekhi, and Farid Alborzi for extreme kindness, heartfelt friendship, and lifting my spirits when I needed it the most. I thank Shahrzad Azizzadeh, Zohreh Asgharzadeh, Katy Salamati, Nastaran Gharkholonarehe, Moin Ayazifar, Reza Jafari, Ramin Shahriari, and Behnam Adib for their blissful company and all the joyful memories.

Above all, my deepest expression of appreciation goes to my family whose ceaseless support and endless love gave me the strength and the confidence to complete this journey. I thank my wonderful sisters, Sara and Sepideh, for encouraging me all the time and being my best friends. Their presence has filled my life with a sense of warmth and bliss. I am extremely grateful to my mother for teaching me to be a better person in my life and to never give up following my dreams. There is no doubt that without her sacrifices, I would not be able to get where I am now. I am forever indebted to her, and words fail me to express my great appreciation of her pure love and endless support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Elastic Optical Networks (EONs)

1.1 Introduction and Related Work

Optical networking technologies are crucial to the operation of the global Internet and its ability to support critical and reliable communication services. In response to rapidly growing IP traffic demands, 40 and 100 Gbps line rates over long distances have been deployed, while there is substantial research and development activity targeted to commercializing 400 and 1000 Gbps rates [1]. On the other hand, emerging applications, including IPTV, video-on-demand, and inter-datacenter networking, have heterogeneous bandwidth demand granularities that may change dynamically over time. Accordingly, mixed line rate (MLR) networks [2] have been proposed to accommodate variable traffic demands.

Nevertheless, optical networks operating on a fixed wavelength grid [3] necessarily allocate a full wavelength even to traffic demands that do not fill its entire capacity [4]. This inefficient utilization of spectral resources is expected to become an even more serious issue with the deployment of higher data rates [5, 6]. Figure 1.1 compares fixed IUT grid versus flexible grid optical networks. As it can be seen there, the fixed grid optical networks can only support data rates of 10, 40, and 100 Gbps with a great portion of bandwidth capacity left unutilized, whereas flexible grid optical networks can also support 400 and 1000 Gbps rates [7]. Still, some parts of bandwidth capacity are unusable to transmit data in flexible grid optical networks, in that the traffic demands associate with them cross the established grids (i.e. black bar).

Elastic optical networks [7, 8] have the potential to overcome the fixed, coarse granularity of existing WDM technology and are expected to support flexible data rates, adapt dynamically to variable bandwidth demands by applications, and utilize the available spectrum more efficiently [6]. The enabling technology for such an agile network infrastructure is orthogonal frequency division multiplexing (OFDM), and other efficient transmission techniques including Nyquist WDM and low-density parity-check (LDPC) based transmission [7]. OFDM, a modula-

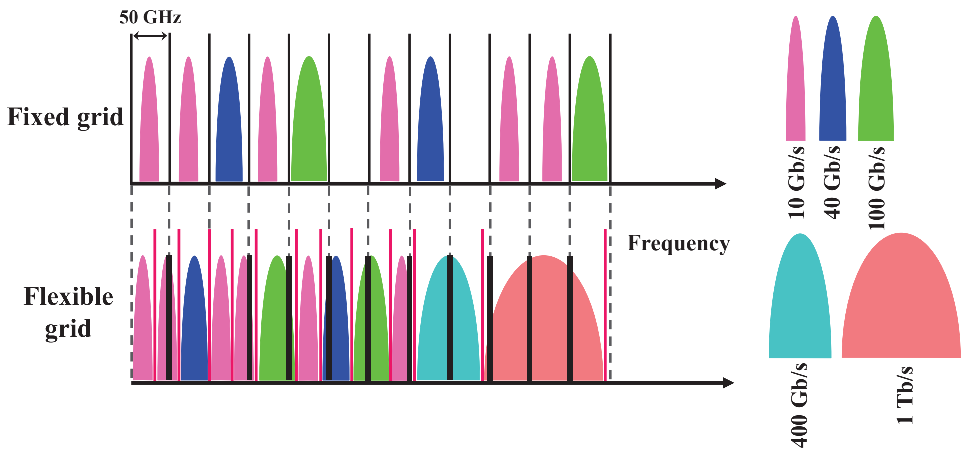


Figure 1.1: Fixed ITU grid versus flexible grid

tion format that has been widely adopted in broadband wireless and copper-based communication systems, is a promising candidate for high-speed (i.e., beyond 100 Gbps) optical transmission [9]. Other key technologies include distance-adaptive modulation, bandwidth-variable transponders and flexible spectrum selective switches; for a recent survey of optical OFDM and related technologies, and how they impact network and control algorithm design, we refer the reader to [9].

OFDM is a multiple-carrier modulation scheme that splits a data stream into a large number of sub-streams [10]. Each data sub-stream is carried on a narrowband sub-channel created by modulating a corresponding carrier with a conventional scheme such as quadrature amplitude modulation (QAM) or quadrature phase shift keying (QPSK). The modulated signals are further multiplexed by frequency division multiplexing to form what is referred to as multicarrier transmission. The composite signal is a broadband signal that is more immune to multipath fading (in wireless communications) and intersymbol interference. The main feature of OFDM is the orthogonality of subcarriers that allows data to travel in parallel, over sub-channels constituted by these orthogonal subcarriers, in a tight frequency space without interference from each other. Consequently, OFDM has found many applications, including in ADSL and VDSL broadband access, power line communications, wireless LANs (IEEE 802.11 a/g/n), WiMAX, and terrestrial digital TV systems.

In recent years, OFDM has been the focus of extensive research efforts in optical transmission and networking, initially as a means to overcome physical impairments in optical communications [11,12]. However, unlike, say, in wireless LANs or xDSL systems where OFDM is deployed as a transmission technology in a *single link*, in optical networks it is being considered as the technology underlying the novel elastic network paradigm [6]. Consequently, in the quest for a truly agile, resource-efficient optical infrastructure, *network-wide spectrum management* arises as the key challenge to be addressed in network design and control.

1.2 OFDM-Based Elastic Optical Networks

OFDM technology is the foundation of the elastic optical network (EON) concept [7], also referred to as "spectrum-sliced elastic optical path network" or *SLICE* [13]. The major difference between RWA and RSA lies in the SLICE network architecture as it flexibly adjusts to the format of the modulation [14]. The main driver of the EON architecture is the ability to allocate bandwidth at the granularity of an OFDM subcarrier rather than at the coarse unit of a wavelength in a fixed-grid network, using bandwidth-variable and format-agile transponders that may be reconfigured dynamically via software [10]. Optical signals are routed along the path to the destination by multi-granular optical switches that adapt to the data rate and center frequency of incoming channels via software control [15, 16]. Bandwidth-variable transponders and switches make it possible to support efficiently a range of traffic demands, from sub- to super-wavelength, by *slicing off* just a sufficient amount of spectral resources along end-to-end paths to satisfy the client requirements. OFDM-based EONs have several advantages relative to existing WDM networks, including [7, 9]:

- *Resilience to physical impairments.* Since each subcarrier operates at a low symbol rate, inter-symbol interference is reduced and the effects of physical impairments are alleviated.
- *Elastic data rates.* The number of allocated subcarriers and the modulation format may be adjusted dynamically, on a per-connection basis, to account for: 1) demand granularity, making it possible to support data rates from Gbps to Tbps, 2) path distance, so as to trade off spectrum utilization for reach, and 3) the time-varying nature of demands. Therefore, EONs may support multiple data rates, either by grouping together any number of subcarriers, as shown in Figure 1.2, or by supporting a different data rate per subcarrier depending on network conditions. Importantly, EONs are highly scalable in that a transition to higher data rates would not require major changes in system design
- *Spectral efficiency.* Two features of OFDM enable highly efficient use of spectral resources. On one hand, adjacent subcarriers may overlap in spectrum due to their orthogonality, as shown in the bottom part of Figure 1.2. This reuse of spectrum increases the overall system capacity. On the other hand, adapting the data rate to demand size, path length, and time variations, achieves better use of existing spectrum.

While the finer granularity of bandwidth allocation is the key feature that makes OFDM attractive for future optical networks, it also introduces new and formidable challenges in

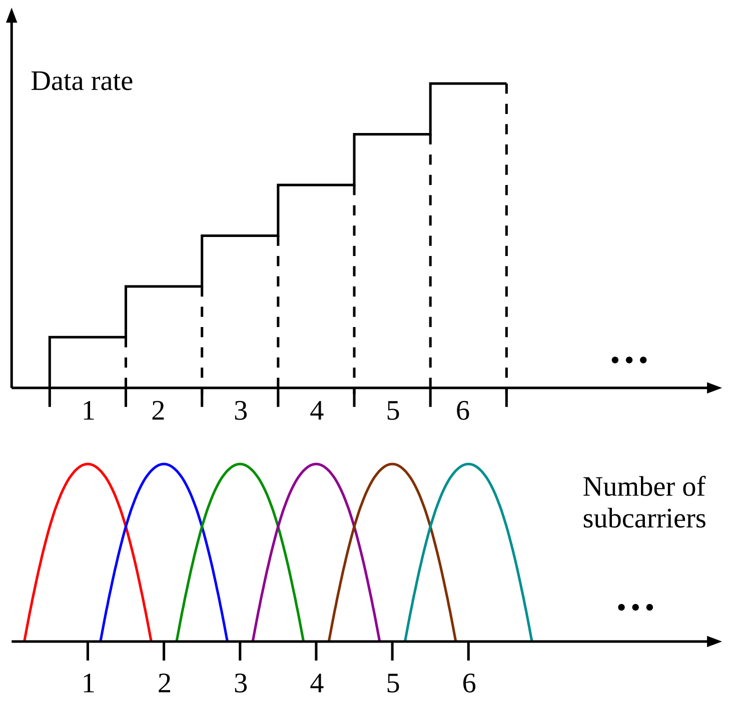


Figure 1.2: Elastic data rates as a function of the number of allocated subcarriers

the design and control of such networks. These challenges call for new spectrum management techniques that address effectively and efficiently issues related to:

- *Scalability.* With an OFDM subcarrier as the unit of bandwidth allocation, the number of spectral resources to be managed network-wide is significantly larger than the number of wavelengths in existing WDM networks.

- *Spectrum contiguity.* If a demand requires t units of spectrum, then t *contiguous* subcarriers must be allocated to it. This constraint, not encountered in wavelength-based networks, may, unless appropriately accommodated, lead to severe fragmentation of the spectral resources that counters the inherent efficiency of fine-grain allocation. Consider Figure 1.3 to see how adjacent subcarriers slots are allocated for each traffic demand.

- *Spectrum continuity.* The same t contiguous subcarriers must be allocated on each link along the end-to-end path of a demand. This constraint is analogous to the wavelength continuity constraint in WDM networks, and further contributes to potential spectrum fragmentation across the network links. As it can be seen in Figure 1.3, if a traffic demand requires to traverse multiple links, then subcarriers associated with this traffic demand occupy the same indices on each link.

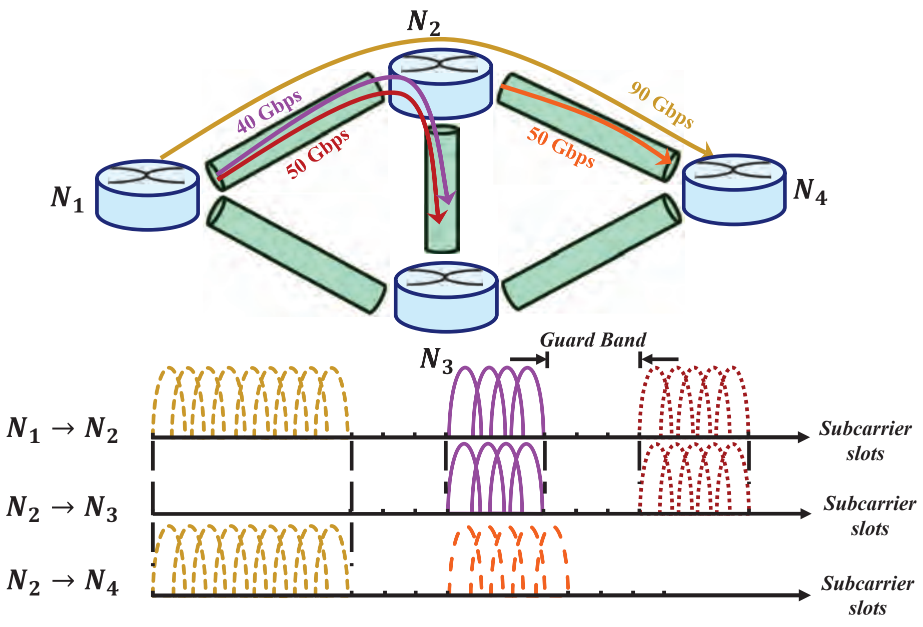


Figure 1.3: Spectrum contiguity and continuity features in EONs

- *Variable data rates.* Support for elastic data rates, a core feature of EONs, requires precise tracking of the spectral width and center frequency of optical signals, and tight coordination of bandwidth-variable transponders and switches along end-to-end paths.

Due to the spectrum continuity constraint, there is a tight coupling between spectrum allocation and routing of a demand. Consequently, routing and spectrum assignment (RSA) [17, 18] has emerged as the essential problem for spectrum management in EONs. Since the performance of a network depends not only on its physical resources (e.g., transponders, physical links, usable spectral width, optical switches, etc.) but also on how it is controlled, the objective of an RSA algorithm is to achieve the best possible performance within the limits of physical constraints. The RSA problem can be cast in numerous forms. The different variants of the problem, however, can be classified under one of two broad versions: *offline RSA*, whereby the traffic demands are known in advance, and *online RSA*, in which a sequence of client requests arrive in some random fashion. Spectrum management techniques for both offline and online RSA are discussed in Chapter 2.

1.3 Structure of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we review and classify recent work in spectrum management for EONs. We introduce multiprocessor perspectives for the spectrum assignment (SA) and the RSA problems in general topology networks in Chapter 3. In Chapter 4, we look into the SA problem in chain networks and propose a suit of heuristics algorithms for this problem. Similarly, we analyze the complexity of the SA and RSA problems in ring networks and propose constant ratio approximation algorithms for these two problems in Chapter 5. Then, we develop a suit of heuristic algorithms for ring networks building upon the multiprocessor scheduling perspective in Chapter 6 such that they either make routing and spectrum assignment decisions jointly or separately. Similar to the chain and ring networks, we study the structure of the optimum solutions in mesh networks and propose a fast and efficient heuristic for this problem in Chapter 7. Finally, we discuss future research directions for this problem in Chapter 8.

Chapter 2

Spectrum Management Techniques for EONs

In Chapter 1, we introduced EONs and reviewed all the advantages and challenges associated with it. In this chapter, we classify the RSA problem as the *offline* and *online* problems. More specifically, we explore variants of integer linear programming (ILP) formulations and heuristics for the offline RSA problem in Section 2.1. Then, we classify existing algorithms for the online RSA problem and discuss performance modeling techniques for this problem in Section 2.2. We also examine and categorize solution approaches for distance-adaptive RSA (DA-RSA) and fragmentation-aware RSA (FA-RSA) in Section 2.3 and Section 2.4, respectively. Next, we present traffic grooming techniques for RSA in Section 2.5 and review survivability mechanisms for EONs in Section 2.6. Finally, we study multi-path RSA in Section 2.7.

2.1 The Offline RSA Problem

In the offline RSA problem, the input typically consists of a set of traffic demands, and the objective is to assign a physical path and contiguous spectrum to each demand so as to minimize the total amount of allocated spectrum (either over the whole network or on any link). Offline RSA arises whenever the traffic patterns in the network are reasonably well-known in advance and any traffic variations take place over long time scales. For instance, offline RSA is an effective technique for provisioning a set of semipermanent connections. Since these connections are assumed to remain in place for relatively long periods of time, it is worthwhile to attempt to optimize the way in which network resources (e.g., physical links and spectrum) are assigned to each connection, even though optimization may require a considerable computational effort. We consider the following basic definition of the offline problem.

Definition 2.1.1 (RSA) *Given*

- *a graph $G = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{A}$ the set of arcs (directed links),*
- *a spectrum demand matrix $T = [t_{sd}]$, where t_{sd} is the number of spectrum slots required to carry the traffic from source node s to destination node d, and*
- *k alternate routes, $r^1_{sd}, \ldots, r^k_{sd}$, from node s to node d,*

assign a route and spectrum slots to each demand so as to minimize the total amount of spectrum used on any link in the network, under three constraints:

1. *each demand is assigned contiguous spectrum slots (spectrum contiguity constraint);*
2. *each demand is assigned the same spectrum slots along all links of its path (spectrum continuity constraint); and*
3. *demands that share a link are assigned non-overlapping parts of the available spectrum (non-overlapping spectrum constraint).*

RSA is a generalization of the well-known routing and wavelength assignment (RWA) problem [19]. If a single route for each source-destination pair is provided as part of the input, and each traffic demand is constrained to follow the given route, the RSA problem reduces to the spectrum assignment (SA) problem.

Definition 2.1.2 (SA) *The RSA problem under the additional constraint that all traffic from source s to destination d must follow the given physical path r_{sd}, (i.e., $k = 1$ in the above definition).*

A recent study [20] considered the complexity of the offline SA problem in chain (path) networks, in which no routing decision is involved. Using results from graph coloring theory, it was shown in [20] that the SA problem in paths is NP-hard, and that a $(2 + \epsilon)$-approximation algorithm (where ϵ is an arbitrary real number that approaches zero) for computing the interval chromatic number of an interval graph may be used for solving the SA problem with the same performance bound. The study also extends this algorithm to solve the SA problem in ring networks with a performance bound of $(4 + 2\epsilon)$.

2.1.1 ILP Formulations and Heuristics

Several variants of the RSA problem have been studied in the literature, and take into account various design aspects. Accordingly, a variety of integer linear program (ILP) formulations have been proposed, each tailored to a specific problem variant. Since the problem is intractable, these ILP formulations cannot be solved within a reasonable amount of time for problem instances corresponding to network topologies encountered in practice. Therefore, an array of heuristic algorithms have been put forward to obtain reasonably good solutions efficiently.

Link-based ILP formulations of RSA as a multicommodity flow problem have been studied in [18, 21, 22]. In addition to the spectrum contiguity, spectrum continuity, and non-overlapping spectrum constraints, the formulations also impose guard carrier constraints in allocating frequency slots (i.e., subcarriers) to the traffic demands. These studies consider three main objectives to minimize: the maximum number of subcarriers allocated on any fiber, the maximum subcarrier index allocated on any fiber, and the total number of subcarriers over all fibers. Upper and lower bounds on the optimal solution for ring and mesh networks were presented under both predetermined and non-predetermined routing. The lower bounds were obtained either using cut-set techniques or the even-load method, and the latter was shown to provide tighter bounds. It was also shown that, in ring networks with uniform demands, the lower and upper bounds are tight. Finally, two heuristic algorithms were developed to solve the RSA problem efficiently. The first algorithm, referred to as shortest path with maximum spectrum reuse (SPSR), uses shortest path routing and the first-fit spectrum allocation strategy to assign frequency slots to demands in decreasing order of their size. The second algorithm, balanced load spectrum allocation (BLSA), considers the k shortest paths as candidates for each demand, and selects the one that minimizes the maximum link load, so as to balance the use of spectrum across the network links.

A similar link-based formulation of an RSA variant referred to as the routing, wavelength assignment, and spectrum allocation (RWSA) problem was studied in [23, 24]. In these studies, the network was assumed to support only a given set of line rates (e.g., 10/40/100/400/1000 Gbps), and each line rate requires a predetermined amount of spectrum that does not depend on the path length. The objective was to select a set of line rates for each demand and a corresponding set of lightpaths (one lightpath per line rate) so as to minimize the total spectrum width in the network. For each lightpath, the ILP formulation determines the center wavelength, the spectral width, and the physical route across the network. It was shown that RWSA is a generalization of the RWA problem in mixed line rate (MLR) optical networks. Due to the complexity of the problem, the ILP formulation was solved only for a six-node network in [23]. Consequently, the problem is decomposed in [24] into two subproblems: (1) the line rate selection problem that determines the line rates for each demand, and which is solved using dynamic programming, and (2) a variant of the routing and wavelength assignment (RWA) problem that seeks to establish lightpaths at the specific line rates. The latter subproblem is solved using three greedy heuristic methods.

A different, path-based ILP formulation of the RSA problem was presented in [17]. In a path-based formulation, k paths, where k is a small integer, are pre-computed for each demand, such that the demand may be routed only along one of these paths. The ILP is used to assign one of the predetermined paths to each demand, while also satisfying the spectrum contiguity and non-overlapping spectrum constraints; the spectrum continuity constraint is implicitly satisfied since the assignment of subcarriers is along a whole path. In [17], the objective is to minimize the number of subcarriers that are used in any link in the network. Although this path-based formulation is more compact than the link-based ILP of [18], the number of decision variables and constraints is substantially large that it cannot be solved directly. Accordingly, a heuristic algorithm called adaptive frequency assignment with collision avoidance (AFA-CA) was presented to select the path for each demand. The algorithm uses a link metric that captures the number of subcarriers that may potentially be allocated to a link, based on the paths that use this link. Then, it processes demands in an adaptive order that depends on previously allocated demands and the current usage of frequency slots in the network, and uses the link metric above to avoid selecting paths that will result in congested links.

Another path-based ILP formulation for the RSA problem was presented in [25, 26], where the objective is to minimize the maximum subcarrier index assigned on any link in the network. In addition to the joint formulation, the study in [26] also presents a decomposition into two formulations that address the routing (R) and spectrum assignment (SA) aspects of the problem, respectively, and is referred to as R+SA. The first formulation takes as input a predetermined set of paths for each demand, and selects one path per demand so as to minimize the amount of traffic flow (i.e., spectrum use) on any link. In the second phase, the SA formulation assigns frequency slots to demands so as to minimize the maximum subcarrier index. While each ILP formulation in the decomposition is more compact and scalable than the joint ILP, such a sequential solution is not guaranteed to yield an optimal solution to the joint problem. A greedy heuristic algorithm is also presented, that processes demands in decreasing order of either their size or their shortest path length; this order is fixed, unlike the heuristic in [17] that adapts the order as the algorithm progresses. A simulated annealing meta-heuristic that builds upon the greedy algorithm was also presented. Finally, the spectrum utilization in an elastic network was compared to that in a fixed-grid WDM network in [25].

The above approaches model the spectrum contiguity constraint by including a set of problem constraints that increase significantly the complexity of the ILP formulation. To overcome this difficulty, the concept of a "channel" as a set of contiguous subcarriers of a given width, was introduced in [27]. For a given spectrum width t, all possible channels with t subcarriers (i.e., one starting at the first subcarrier, one starting at the second subcarrier, and so on) are defined on each link of the network. Then, the RSA problem is transformed to one of "routing and channel allocation," in which channel assignment implies allocation of contiguous spectrum, and no explicit spectrum contiguity constraints are needed. The result is a more compact formulation that achieves a significant speed-up in running time compared to those that directly account for the spectrum continuity constraints [27].

2.2 The Online RSA Problem

Under a dynamic traffic scenario, clients submit to the network requests for optical paths to be set up as needed. Thus, connection requests are initiated in some random fashion. Depending on the state of the network at the time of a request, the available spectral resources may or may not be sufficient to establish a connection between the corresponding source-destination node pair. The network state consists of the physical path (route) and spectrum assignment for all active connections. The state evolves randomly in time as new connections are admitted and existing connections are released. Thus, each time a request is made, an algorithm must be executed in real time to determine whether it is feasible to accommodate the request, and, if so, to perform routing and spectrum assignment. If a request for a connection cannot be accepted because of lack of resources, it is blocked. Therefore, the blocking probability of connection requests arises as the key performance metric of interest in an online RSA scenario.

2.2.1 Heuristic Algorithms

Because of the real-time nature of the problem, RSA algorithms in a dynamic traffic environment must be simple and fast. Since combined routing and spectrum assignment is a hard problem, most studies devise heuristic algorithms. Online RSA algorithms can be broadly classified in two categories depending on whether they tackle the routing and spectrum assignment aspects jointly (i.e., in one step) or separately (i.e., in two steps).

2.2.1.1 Two-Step Algorithms

Two-step online RSA heuristics decompose the problem into two subproblems, the routing problem and the spectrum assignment subproblem, which are then solved sequentially. Specifically, such an approach consists of the following steps:

- Compute a number of candidate physical paths for each source-destination node pair and arrange them in a path list.
- Starting with the path at the top of the corresponding list, use a spectrum allocation policy to assign a feasible path and set of contiguous slots, if they exist, for the requested connection.

The routing algorithm may be *static*, in which case the paths are computed and ordered offline, or *adaptive*, in that the paths computed and their order may vary according to the current state of the network. The number of path choices is another important parameter. Often, as in [28], multiple alternate paths are computed for each request. The spectrum allocation policy determines which set of available contiguous slots are assigned to a request, and is crucial to the performance of an online RSA algorithm. A *first-fit* policy [28, 29] selects the lowest index set, a *random-fit* policy [30] randomly allocates one of the available sets, whereas a *best-fit* policy selects the smallest set that can satisfy the request. Note that an improvement in the operation of the first-fit policy has been proposed in [31]. This study 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 performs far better than the conventional first-fit policy.

The study in [29] investigated the optimal slot width for EONs by measuring the blocking probability under dynamic traffic using Monte Carlo simulations. Each demand was routed on its shortest path, and the first-fit policy was used for spectrum allocation. The main finding of the study was that the best performance is achieved when the slot width is equal to the greatest common factor of the spectrum widths of the data rates supported in the network. On the other hand, the KSP-based RSA algorithm proposed in [28] first computes k shortest paths for a given request, and then searches each path to find the required number of contiguous subcarriers using the first-fit policy.

The two-step online RSA algorithm in [32] proposes an interesting adaptive routing algorithm to solve the first subproblem. Specifically, routing tables at each node associate a probability with each (next hop, destination) pair. A dynamic ant colony optimization (ACO) algorithm is run continuously that updates these probabilities based on spectrum usage information on each link collected by ants as they traverse the network. When a request arrives, a path is selected starting at the source and visiting next-hop nodes based on the current values of the probabilities stored at the routing tables until the destination has been reached. Once the path has been determined, the first-fit policy is used to assign frequency slots to the request. This ACO-based online RSA algorithm compares favorably to the KSP-based algorithm in [28].

2.2.1.2 One-Step Algorithms

One-step online RSA heuristics, on the other hand, solve the two subproblems (i.e., routing and spectrum assignment) simultaneously but sub-optimally, typically using a greedy approach. In [33], a dynamic version of the RWSA problem studied in [23, 24] is considered, and is referred to as D-RWSA. As with RWSA, D-RWSA was decomposed into a rate selection problem (similar to the one we discussed earlier), and a dynamic routing and channel selection problem that is a variant of online RSA (in this work, the notion of a channel is similar to that defined in [27]). This latter problem was solved using an auxiliary graph that represents the state (i.e., established connections) of the network. With this representation, finding a path and contiguous spectrum for a new request reduces to finding a shortest path on the auxiliary graph.

Two different one-step online RSA algorithms were introduced in [28]. In the first, a modified Dijkstra shortest path algorithm is employed, in which, every time a link is considered to extend the shortest path to the destination node, the link is checked to ensure that it has available contiguous spectrum in common with links already in the path. If so, the link is added, otherwise the algorithm considers other links even if they have a higher cost. The second one-step RSA algorithm builds a path vector tree routed at the source node such that all links along each path have sufficient contiguous spectrum for the request. The algorithm then searches the path vectors to find one that has available spectrum and minimum cost; this algorithm is more computationally demanding but results in better blocking performance. A modified Dijkstra algorithm similar to the one in [28] was proposed in [34] to solve the online RSA problem in one step. The main difference is that the spectrum availability along the path is maintained by keeping track of the channels (as defined in [27]) that may fit the given connection request.

2.2.2 Performance Models

Evaluating the performance (e.g., in terms of blocking probability, spectrum utilization, or fragmentation rate) of an EON operating under a given online RSA algorithm requires analytical models that are able to capture accurately the evolution of the system's state. However, developing such models is a challenging task due to the nature of the RSA problem. The spectrum continuity constraint introduces load correlation between subcarriers in adjacent links, similar to the wavelength continuity constraint in current WDM networks. In addition, the spectrum contiguity constraint introduces correlation between subcarriers in the same link. Hence, it is not possible to use multiclass $M/M/K/K$ models directly, as these do not account for the fact that a new connection must occupy a set of *contiguous* servers simultaneously.

An exact analytical model for a *single* link of an EON has been developed in [30]. The model captures three important features of such a system: the load correlation among subcarriers due to the contiguity requirement; the fact that subcarriers are not equivalent (e.g., the lowest and highest index subcarriers are not equivalent to other subcarriers since they have contiguous subcarriers on one side only); and the existence of multiple classes of calls, each requiring a different spectral width. Under the assumption of Poisson arrivals, two continuous-time Markov chain models were developed, one each for the first-fit and random-fit allocation policies. Since the state space grows exponentially with the number of subcarriers, two heuristic algorithms were also developed to compute the stationary distribution from which the blocking probability and other parameters may be obtained.

Exact and approximate Markov chain models for a special version of a spectrum allocation problem are developed in [35]. This study considers demands with time-varying requirements and proposes spectrum extraction and contraction policies to adjust dynamically the amount of spectrum assigned to each demand in response to these requirements. The analytical models are used to evaluate the blocking performance of the various policies. These blocking models are then used within an iterative RSA algorithm to minimize the average blocking probability.

2.3 Distance-Adaptive RSA (DA-RSA)

The design of commercial WDM networks has traditionally focused on optimizing the transmission performance in the worst case [36]. Specifically, for a given data rate (e.g., 40 Gbps), the modulation format is determined so as to ensure that the signal is transmitted with sufficient quality along the worst path (typically, the longest path and/or the one with the most hops) in the network. Then, the same format is used for every, say, 40 Gbps demand. In other words, each such demand is assigned the exact same amount of optical spectrum irrespective of its path length, the number of nodes it traverses, or the level of impairments it encounters. Engineering for the worst-case scenario leads to low spectrum utilization given that the transmission quality along most paths is far better than that in the worst case.

In OFDM-based networks, on the other hand, it is possible to adjust the modulation format and/or number of bits per symbol to account for link impairments (e.g., available signal-to-noise ratio (SNR)) so that demands with the same data rate are allocated different amounts of spectral resources depending on the quality of their path [10, 37--41]. Distance-adaptive (DA) spectrum allocation, a concept first introduced in [36], exploits the tradeoff between spectrum width and reach (for the same data rate) to improve utilization [42] by tailoring the modulation format to the level of impairments: a high-level modulation format with narrow spectrum and low SNR tolerance may be selected for a short path, whereas a low-level modulation with a wider spectrum and high SNR tolerance may be used for a longer path [43]. In fact, it has been argued [36] that the utilization of spectral resources depends not only on the network size and topology, node and link characteristics, or traffic pattern, but also on the specific RSA algorithm employed.

An offline version of the DA-RSA problem, referred to as the routing, modulation level, and spectrum allocation (RMLSA) problem was studied in [44]. In this problem, each demand is 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. A path-based ILP formulation for RMLSA was first provided, and then the problem was decomposed into two subproblems, routing and modulation level (RML) and spectrum assignment (SA) and solved sequentially (RML+SA) using ILPs. Finally, a greedy heuristic and a simulated annealing meta-heuristic, similar to the ones in [26], were also presented.

Most studies of DA-RSA consider the online version of the problem and develop heuristic algorithms to accommodate randomly arriving connection requests. Several algorithms follow a two-step approach similar to the one we discussed in Section 2.2.1.1 for the basic (i.e., non-distance-adaptive) online RSA problem. The main difference is that a spectrum allocation model is used to determine the number of subcarriers as a function of data rate and path length. Therefore, in the second step of the algorithm, the number of slots that the spectrum allocation policy must search for varies depending on the length of the path considered.

Two-step heuristics for the online DA-RSA problem were presented in [36, 45]. These algorithms compute a number of fixed-alternate paths (i.e., they use static routing) for each source-destination pair, and order them in decreasing length. In the second step, they employ the first-fit spectrum allocation policy and sequentially consider each path until a number of contiguous frequency slots that can accommodate the requested data rate over the given path length is found; if no spectral resources are available on any of the paths in the list, the request is blocked. The main difference between the two algorithms is that the latter only considers paths for which the number of slots is the same as for the shortest path. A similar algorithm was used in [39] for routing of super-wavelength demands.

A version of the online DA-RSA problem referred to as dynamic impairment aware routing and spectrum allocation (IARSA) was studied in [43]. The objective was to select a feasible path for each request and allocate subcarriers by using an appropriate modulation format with the transmission reach for the requested data rate. Two variants of IARSA were investigated, one in which regenerators may modify the modulation format of the incoming signal, and one in which the modulation format does not change in the network. The heuristic algorithm used to solve the IARSA problem works as follows. For each modulation format, each link in the network is assigned a weight equal to the ratio of the required spectrum over the number of free slots on this link. For each modulation format, a modified version of Dijkstra's algorithm is used to find a minimum cost path with sufficient contiguous spectrum for this request; this approach is similar to the one-step heuristics for the basic online RSA problem we discussed in Section 2.2.1.2. Finally, the path and modulation format with the smallest cost (if any is found) is assigned to the request.

A quality of transmission (QoT) aware online RSA technique was proposed in [14], consisting of three stages: path calculation, path selection, and spectrum assignment. The Dijkstra and k-shortest algorithms were adapted for computing paths, while fiber impairments and non-linearity effects at the physical layer were modeled using a closed-form expression that estimates the QoT along a given path. For each request, the most feasible route is chosen with respect to optical signal-to-noise ratio (ONSR), as determined by the physical layer model.

2.4 Fragmentation-Aware RSA (FA-RSA)

Whereas the ability to allocate variable data rates is an attractive feature of EONs in terms of supporting heterogeneous applications, two challenges arise in operation when connections arrive and depart dynamically [46]:

- *Fragmentation.* Fragmentation of spectral resources emerges as allocation and de-allocation of blocks of contiguous slots on demand may cause part of the spectrum to become unusable. There are two sources of stranded spectrum in EONs [46]. The spectrum continuity constraint causes *horizontal* fragmentation in that the same block of spectrum may not be available along successive links of a path despite the fact that each link may have sufficient bandwith for a request; this issue is similar to wavelength fragmentation in WDM networks. Variable data rates along with the spectrum contiguity constraint, on the other hand, are the cause of *vertical* fragmentation, a situation whereby the spectral resources on a single link are fragmented into small non-contiguous blocks that cannot be allocated to a single large demand.

- *Fairness.* If spectral resources become fragmented across the network links, heterogeneous connection requests will experience blocking rates that depend strongly on their data rate and/or path length. Due to vertical fragmentation, large contiguous blocks of slots may become sparse, hence high-rate connections will be more likely to be rejected than low-rate ones. On the other hand, for a given amount of spectrum, horizontal fragmentation makes it more difficult to find continuous blocks on long paths compared to short ones. Therefore, if left unchecked, fragmentation may lead to starvation of high-rate and/or long-path connections.

Fragmentation-aware RSA algorithms attempt to improve the blocking performance, fairness, and spectrum utilization of EONs by minimizing the extent of spectrum fragmentation. Figure 2.1 schematically tries to depict how these algorithms are working. Figure 2.1(a) shows that the number of subcarrier indices could be reduced if another path with sufficient number of slots is taken to transmit a traffic demand. On the other hand, Figure 2.1(b) displays that defragmentation can be performed by squeezing all the subcarriers with regard to SLICE features (i.e. spectrum continuity and contiguity constraints).

The first step in such an approach is to develop metrics that quantify the degree of fragmentation in the network. The utilization entropy concept introduced in [47] assesses the spectrum fragmentation of a link by counting the number of neighboring pairs of slots that have different status (i.e., one slot is used but the other is free), and normalizing this number to get a value in $(0, 1)$ such that low (respectively, high) values represent low (respectively, high) vertical fragmentation. A similar concept was defined to capture horizontal fragmentation along

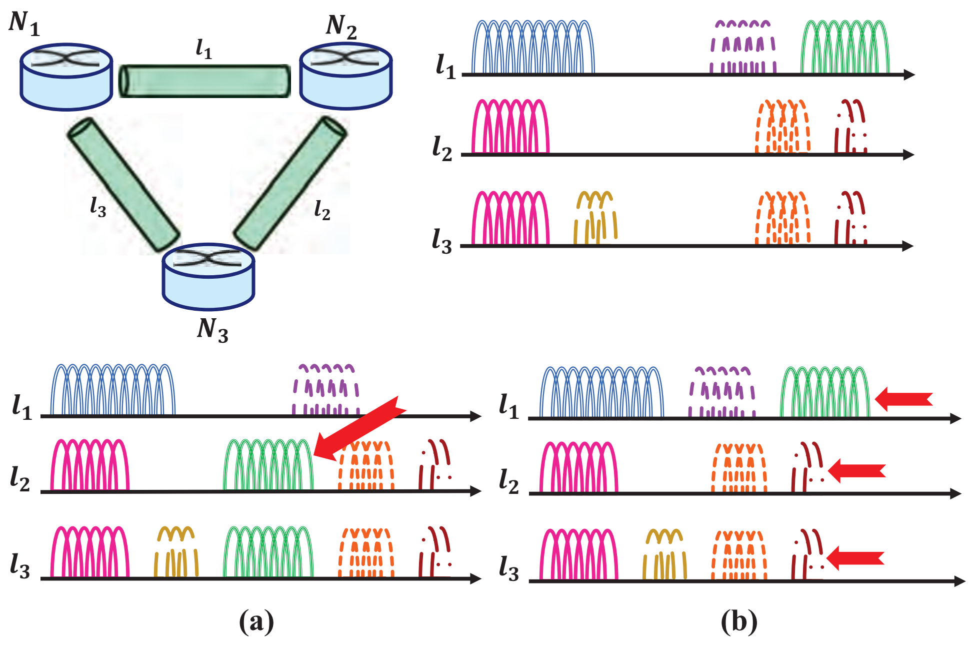


Figure 2.1: Defragmentation in EONs; a) moving subcarriers; b) shifting subcarriers

a path by considering the status of a given slot on pairs of successive links in the path. The fragmentation ratios proposed in [48, 49] to measure the bandwith fragmentation of a link or path are inspired by similar ratios proposed for storage systems, and take into account not only the status of neighboring slots or that of a slot on adjacent links, but also the size of free blocks of slots. In [46], each contiguous block of unused slots is assigned a value equal to the maximum data rate that it can support, and the fragmentation index of a path is computed by taking the ratio of the sum of the value of available blocks to the value of the sum of the slots in all blocks; the latter sum represents the value of these slots as if they were contiguous. The fragmentation index concept is extended to capture spectrum fragmentation over the whole network by taking the average of the fragmentation indices of the shortest paths between every source-destination pair.

Fragmentation-aware RSA algorithms may be classified as *proactive* or *reactive* [46], as we discuss next.

2.4.1 Proactive FA-RSA

Proactive FA-RSA techniques attempt to prevent or minimize spectrum fragmentation at the time a new request is admitted to the network. Since support for variable data rates is a main contributor to fragmentation, [46] identifies four spectrum management techniques for allocating spectrum to connections of different data rates. With "complete sharing," all connections share the whole spectrum using the first-fit policy. With ‘‘pseudo partition,’’ low-rate connections are allocated bandwidth from one end of the spectrum (using first-fit), while high-rate connections are allocated from the other end (again using first-fit). Under the ‘‘dedicated partition’’ scheme, spectrum is partitioned and each partition is dedicated to serving connections of a given data rate; hence, vertical fragmentation due to variable rates is eliminated and each partition reduces to a wavelength-routed network. Finally, ‘‘shared partition’’ is a generalization of the previous scheme in that spectrum is partitioned but higher data rate connections may access partitions assigned to lower data rate ones. It was shown in [46] that the dedicated and shared partition schemes improve both fairness and fragmentation compared to complete sharing or pseudo partition.

A similar concept of spectrum reservations is studied in [50]. Instead of partitioning spectrum such that each partition is shared only among connections with the same rate, [50] proposes that a block of contiguous subcarriers be reserved for each source-destination pair. In addition, subcarriers that are not reserved may be shared on demand among all connections. This study assumes that demands vary with time, but as long as a demand stays within its reservation, it can always be accommodated. However, if a connection requires additional bandwidth, an FA-RSA algorithm is executed to allocate shared subcarriers along one of a set of candidate paths.

Two FA-RSA algorithms that make spectrum allocation decisions based on the current state of fragmentation were introduced in [49]. The first algorithm assigns each new request to a path that minimizes a network-wide fragmentation ratio defined in the same study. The second algorithm attempts to utilize slots that are already used the most in the network. The maximize common large segment (MCLS) algorithm in [51] generates a number of candidate paths for a request. For each candidate path, the algorithm considers all the links in the path (i.e., candidate links), as well as links that are adjacent to the candidate links at any node on the path. The algorithm then computes a metric that captures the availability of contiguous slots between candidate and adjacent links after spectrum for this request has been allocated. Finally, it selects the path and spectrum allocation that has the smallest value for this metric, so as to maximize the probability that future requests will find a sufficient number of contiguous slots to be accepted.

We also note that all the FA-RSA algorithms above [49--51] can be classified as two-step heuristics based on our discussion in Section 2.2.1.1.

2.4.2 Reactive FA-RSA

Recognizing that fragmentation may not be completely eliminated in a dynamic environment, reactive FA-RSA algorithms employ defragmentation techniques to restore the network’s ability to accommodate high-rate and long-path connections. The objective of defragmentation is to rearrange the spectrum allocation of existing traffic demands so as to consolidate available slots into large contiguous and continuous blocks that may be used to establish future requests. Defragmentation strategies may be broadly classified as *periodic* or *path-triggered* [34]. Periodic defragmentation runs at long time scales and is initiated either at regular intervals or whenever a metric indicative of network-wide fragmentation exceeds a certain threshold. The process tackles the fragmentation of spectral resources across the whole network, and hence it is computationally expensive and may disrupt ongoing connections along large parts of the network. The scope of path-triggered defragmentation, on the other hand, is more narrow. It is invoked when the online RSA algorithm is unable to find adequate resources to satisfy a new traffic demand, with the objective of assembling continuous/contiguous blocks of spectrum sufficient for this demand. Therefore, path-triggered defragmentation is executed at shorter time scales but only disrupts connections sharing a link with the new request. With either strategy, a make-before-break rerouting (MBBR) [40] scheme may be used to minimize the disruption to existing connections as they are moved to a new spectrum block. Spectrum defragmentation has been demonstrateed experimentally in [52].

The network-wide defragmentation problem was studied in [53] under two objectives. The problem was defined as one of rearranging existing connections so as to minimize the total spectrum required for these connections (and, hence, maximize spectrum consolidation for future connections). However, this objective may be achieved with an offline RSA approach, and does not take into account the disruption to connections caused by the rearrangement. Therefore, the study also considered the secondary objective of minimizing the number of connections interrupted during the reconfiguration process. The problem was formulated as an ILP, and two heuristics were proposed to solve it.

A path-triggered defragmentation strategy was proposed and investigated in [34]. When a new request arrives, an online RSA algorithm is first run to accommodate it. If a path with sufficient spectrum cannot be found, then a defragmentation algorithm is triggered that consists of the following steps. First, the number of available (but not necessarily continuous or contiguous) slots along each of a set of candidate paths is computed. If a path with a sufficient number of free slots does not exist, then the request is blocked. Otherwise, an attempt is made to create a block of continuous and contiguous slots along the shortest path with sufficient available resources. To this end, connections that share links with this shortest path must be reallocated a different block of spectrum. The problem of identifying the connections to be rearranged and the new spectrum allocation (for the old connections and the new request) was formulated as an ILP. The objective was to minimize the number of connections to be rearranged, under the constraint that the routes of existing connections are not modified. In order to solve the problem for large networks, a greedy randomized adaptive search (GRASP) meta-heuristic was proposed.

The spectral defragmentation algorithm in [54] consists of two phases. When a new request arrives but cannot be accommodated, a greedy heuristic is executed in the first phase to identify the spectrum block for the new request that minimizes the number of ongoing connections that will be affected. In the second phase, the problem of assigning new spectrum to the affected connections is formulated as a bipartite matching problem. If a perfect matching (i.e., one that satisfies all connections) exists, then existing connections are moved to the new spectrum and the request is assigned the spectrum block selected in the first phase; otherwise, the request is blocked or the algorithm repeats from the first phase to determine a new spectrum block for the request.

A set of reactive defragmentation strategies that capitalize on hitless optical path shift (HOPS) were described in [55]. HOPS technology, assessed in [56] where it is referred to as ‘‘push-pull defragmentation,’’ allows a connection to shift to a new block of spectrum as long as the route of the connection does not change and the move to the new spectrum does not affect other established connections. A simple technique was proposed in [55] to consolidate the spectrum freed by a terminated connection with other blocks of spectrum available along the links of its path. Specifically, upon the departure of a connection, ongoing connections that share a link with the just terminated connection are shifted, whenever possible, to the lower end of the spectrum, thus creating larger available blocks at the higher end for future connections. A more complex algorithm was proposed to create continuous and contiguous blocks of spectrum for new requests whenever they cannot be accommodated by a basic online RSA algorithm that is applied first.

2.5 Traffic Grooming with RSA (TG-RSA)

Traffic grooming [57] is an optimization problem that arises in the design and control of networks with multigranular traffic. In conventional WDM networks, the main objective is to aggregate sub-wavelength demands onto lightpaths of fixed capacity so as to improve the utilization of wavelengths and reduce the number of add/drop ports that represents a major cost for the network [58]. The aggregation of traffic takes place at specific grooming nodes that are analogous to hub airports where passengers are ‘‘groomed’’ onto fixed-capacity airplanes.

In EONs, lightpaths (or ‘‘LambdaFlex connections’’ [59]) may be set up with the exact amount of spectrum to meet the data rate and reach for a traffic demand. Nevertheless, traffic grooming in EONs has the potential to improve spectrum utilization as well as yield significant cost reduction in terms of transponders. Specifically, aggregating demands for transport on a single lightpath presents two opportunities for spectrum savings. First, depending on the modulation format, transporting a single high-rate (e.g., 400 Gbps) connection is generally more

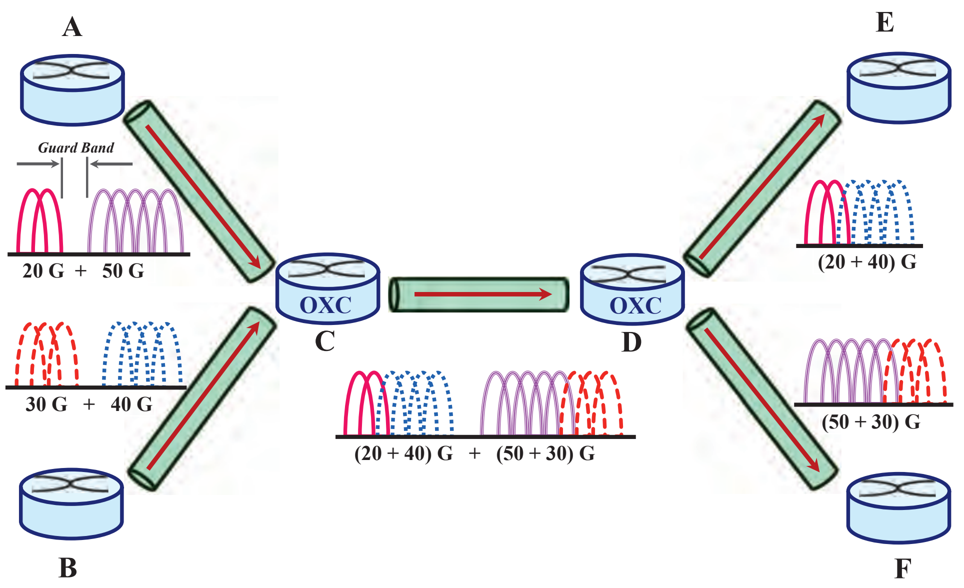


Figure 2.2: Traffic grooming in EONs

spectrum efficient than transporting a number of low-rate connections with the same total data rate (e.g., 4×100 Gbps) [60]. Second, a smaller number of independent lightpaths results in a smaller number of guard bands around these lightpaths, hence reducing the spectrum required to support a given set of demands (or, equivalently, increasing the amount of spectrum available to carry client traffic). Importantly, in EONs it is possible to perform traffic grooming at the optical layer thereby eliminating expensive O/E/O operations. Figure 2.2 illustrates how traffic grooming works in EONs where both nodes A and B send two distinct traffic demands to node C (i.e. $20G$ and $50G$ from A and $30G$ and $40G$ from B). The optical cross-connect (OXC) at node C figures out that incoming traffics of $20G$ and $40G$ must be transmitted to node E, whereas demands of size $50G$ and $30G$ must be sent to node F. Hence, it combines the demands with the same destination into a one such that there is no guard band between them. Finally, an OXC installed at node D routes these combined traffics to their final destinations. Optical traffic grooming involves setting up an optical tunnel [59] that carries several LambdaFlex connections in a contiguous block of spectrum; the bandwidth-variable transceiver and switch requirements to implement such optical tunnels are discussed in [61].

Once grooming considerations are included in the RSA problem (whether online or offline), the new TG-RSA problem becomes significantly more difficult to tackle optimally. Mixed integer linear programming (MILP) formulations of offline TG-RSA variants have been presented in [61--63]. The objective in [61, 62] was to minimize the total amount of spectrum used on any link, whereas the objective function of [63] is an average spectrum utilization rate weighted by fiber length. The formulations in [61, 62] are path-based, and, in fact, the formulation presented in [61] leverages the formulation of the basic offline RSA in [25] and adds grooming-specific constraints. Two TG-RSA heuristics are presented in [61] to solve the problem efficiently on realistic networks. The least spectrum grooming (LSG) heuristic attempts to minimize the spectral resources by grooming demands with the same source and paths that share the most links. The minimum transmitter grooming (MTG) algorithm is similar to LSG but its goal is to minimize the number of transponders.

The online TG-RSA problem has been addressed in [64], and an auxiliary graph was used to model the problem. Specifically, each physical node is represented using several layers in the auxiliary graph: the transponder layer captures the O/E/O conversion (i.e., grooming) ability and the spectrum layers represent the available spectrum resources. The spectrum continuity constraint is enforced by having edges between the same spectrum layers of adjacent nodes. The spectrum contiguity constraint is enforced by deleting edges between spectrum layers if the amount of contiguous spectrum is not sufficient to carry a demand. A path for a demand is found by running a shortest path algorithm on the auxiliary graph representing the state of the network at the time a request arrives, as well as the data rate requirement of the request. This auxiliary graph approach makes it possible to apply a range of grooming policies (e.g., minimize the number of transponders, the number of physical or logical links, etc.) by setting the weights of the edges appropriately.

The auxiliary graph model of [64] was used in [65] to find both a working and a backup lightpath in a dynamic traffic grooming scenario under shared protection. The working path is selected among the k shortest paths in the auxiliary graph, while the backup path is link-joint with the working path. This study proposes a protection scheme referred to as elastic separate-protection-at-connection (ESPAC) that allows the backup lightpaths of two connections to share spectrum if the respective working lightpaths are link-disjoint. ESPAC maximizes the opportunities for spectrum sharing by using the first-fit policy to assign spectral resources, starting with one end of the spectrum for working lightpaths and on the other end for backup lightpaths. Protection and restoration schemes for EONs are discussed in more detail in the following section.

2.6 Survivability and RSA

OFDM lays the foundation for EONs to support individual data rates of 400-1000 GBps [1] and, hence, aggregate throughput per fiber of potentially tens to hundreds TBps. Since the failure of even a single network element (e.g., a fiber link) is likely to affect numerous connections and result in immense data loss, network survivability assumes critical importance. Survivability refers to the ability of the network to reconfigure itself so as to restore the connections affected by a failure. In an optical network, three types of failures are generally considered: link failures (e.g., caused by cable cuts), node failures (e.g., due to equipment malfunction at a switch or router), and channel failures (e.g., caused by the failure of transmitting or receiving equipment specific to that channel).

Several survivability mechanisms have been explored in WDM networks [66] and can be classified in one of two broad categories: *protection* or *restoration*. Protection schemes are carried out at the network design or planning phase in *anticipation* of a network failure. Backup resources reserved by these schemes may be *dedicated* to a single connection or *shared* among multiple connections. During normal network operation, reserved resources remain idle. Upon occurrence of a failure, affected connections are redirected to reserved resources according to the plan determined at the planning phase (typically, at the time a connection was set up). Figure 2.3(a)-(b) illustrate the case where a dedicated path is considered as a backup path (BP) for each connection request (CR), while these BPs may use a common path. However, Figure 2.3(c)-(d) represent shared backup path for several CRs such that these CRs may use the same path as their primary path in EONs.

A restoration scheme, on the other hand, does not reserve resources for a connection at the time of establishment. Instead, resources for recovering a connection are discovered dynamically immediately *after* the occurrence of a failure affecting the connection. With either protection or restoration, a recovery scheme may be either *failure-independent* (also referred to as path-based) or *failure-dependent* (also referred to as link-based). In failure-independent recovery, the source and destination nodes of each affected connection take action to switch to a backup path that is disjoint with the corresponding failed working path, regardless of the location or type of failure. In contrast, with failure-dependent recovery, it is the nodes at either side of the failure (e.g., the endpoints of a failed link) that redirect all affected connections around the failure without intervention of the various source and destination nodes. As was remarked in [67], similar survivability mechanisms may be applied to EONs.

2.6.1 Protection

The problem of ensuring dedicated or shared protection of established connections in an EON may be viewed as a variant of the offline and online RSA problems with additional constraints to account for the backup paths and the sharing (if any) of backup spectrum. The following studies all consider failure-independent mechanisms. Note that the ESPAC algorithm [65] we discussed in Section 2.5 also implements a failure-independent shared-protection scheme in a traffic grooming context.

Two-fiber ring networks with 1+1 dedicated and 1:1 shared protection were considered in [68], and DA-RSA algorithms were proposed to assign spectrum to the working and backup

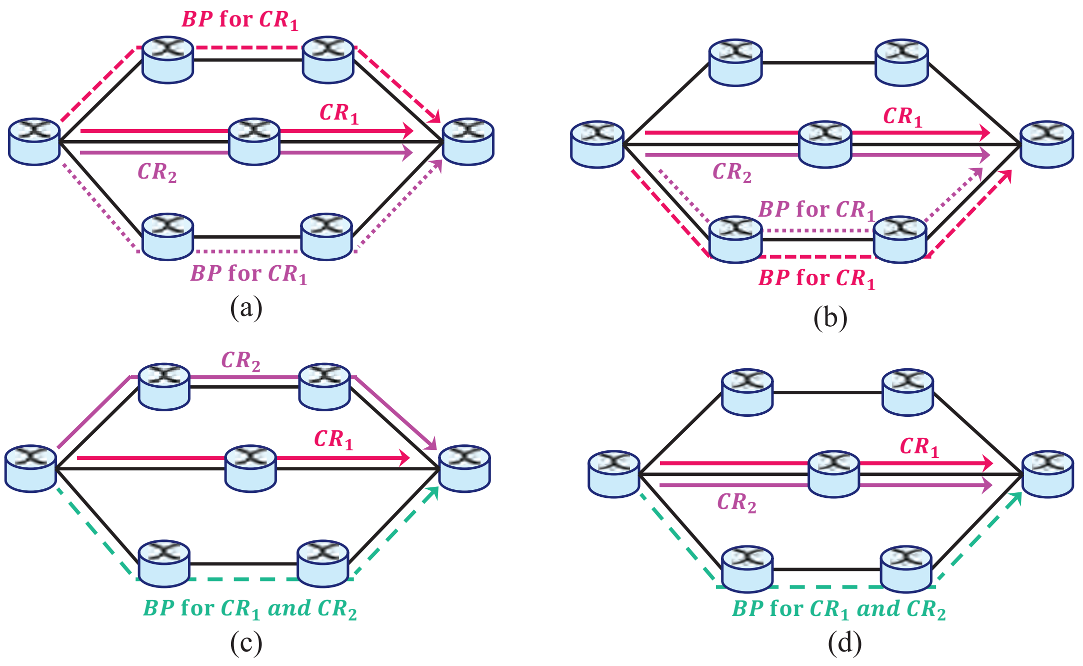


Figure 2.3: Dedicated versus shared path survivability in EONs

paths. In both cases, the shortest (respectively, alternate) path around the ring was assigned as working (respectively, backup). With 1+1 dedicated protection, the same set of frequency slots are assigned to the working and backup paths of a connection independently of other connections. With 1:1 shared protection, on the other hand, the algorithm assigns the same set of slots to the working paths of several connections as long as they are pair-wise disjoint, and similarly for the backup paths of the same set of connections.

The offline RSA problem was considered in [69] under the additional constraint that all demands be assigned two disjoint paths, one working and one backup, and that spectral resources on backup paths be shared among connections with disjoint working paths. A MILP formulation of the problem was developed with the objective of minimizing the amount of spectrum allocated on the most congested link. Furthermore, a heuristic algorithm was proposed. The algorithm finds the k shortest *cycles* for each demand, and processes demands in decreasing order of shortest cycle length. It uses the first-fit policy to assign spectrum for the demand on each cycle (hence, on the working and backup paths), and selects the one that yields the lowest congestion on any link of the cycle.

The online RSA problem with dedicated protection was studied in [70]. The objective in this case is to find a working and backup path for each arriving connection request that are link-disjoint; the two paths are assumed to share transponders, hence they must be assigned the exact same spectrum block. The problem is solved by first constructing a set of auxiliary graphs that represent the network state, and then running Suurballe's algorithm on each graph to find the shortest pair of disjoint paths for the request. The i-th auxiliary graph includes an edge between two nodes if the corresponding fiber link has a number of contiguous slots starting at slot i that may satisfy the demand. The algorithm selects the shortest pair of paths among the ones constructed.

The study in [71] considered the online RSA problem with shared protection, and introduced two policies for spectrum sharing among backup paths. Under the conservative policy, sharing is allowed only if the backup paths have the same bandwidth, whereas under the aggressive policy, sharing is allowed even if the backup paths have different bandwidth. Of course, with the aggressive policy, the number of resources allocated to the backup path equals the maximum bandwidth on the working paths that share this backup path. It is noted that, while the aggressive policy leads to more sharing opportunities, it may fragment the spectrum along backup paths. A heuristic algorithm was proposed that first computes the working path and then the backup path (each selected from respective k shortest paths), and allocates spectrum using the first-fit policy.

A recent study [72] considered three types of networks (single- and multi-rate WDM networks, and EONs) and compared them in terms of cost and energy efficiency under dedicated (1+1 or 1:1) and shared protection. The respective RWA and RSA problems were solved using heuristics. The heuristics are similar to ones discussed earlier in this section in that candidate pairs of working/backup paths for each connection were computed using a k-shortest path algorithm. The main difference is that each candidate pair was evaluated using a metric that accounts for power consumption on both paths.

2.6.2 Restoration

A new survivability scheme called bandwidth squeezed restoration (BSR) was first proposed in [73] and was further refined in [74]. BSR takes advantage of the variable, fine-granularity spectrum allocation possible in EONs, and may be adapted for operation under either a pre-planned protection or a dynamic restoration mechanism. In its original form [73], BSR assumes that the data rate assigned to a connection is the sum of a committed rate and an excess rate. Consequently, recovery does not start until a failure causes the data rate of a connection to fall below the committed rate. At that time, a new backup path is selected (either from an existing protection plan or discovered dynamically) that can support the committed rate of the connection. As part of the recovery process, the bandwidth allocated to some connections that are not affected by the failure (e.g., connections with a working path that overlaps with the backup path of an affected connection) may be *squeezed* to the corresponding committed rate, so as to provide maximum survivability for the given failure.

The BSR scheme was extended in [74] to allow for three recovery scenarios: (i) full bandwidth guaranteed recovery (FBGR), (ii) partial bandwidth guaranteed recovery (PBGR), or (iii) best-effort recovery (BER). FBGR is a conventional recovery scheme, while PBGR is akin to restoring the committed rate as in [73]. The BER scheme is unique to EONs and makes it possible to allocate any amount of spectral resources available at the time of failure so that a connection may proceed even at a low rate. Since the amount of available resources must be discovered after the failure, even if the backup path has been pre-planned, BSR may be classified as a dynamic restoration scheme.

The dynamic load balancing shared-path protection (DLBSPP) algorithm presented in [67,75] is a hybrid protection/restoration algorithm. Similar to shared protection schemes we discussed in the previous subsection, DLBSPP protects from single failures by allocating a working and backup path to each connection at setup time. An interesting feature of the algorithm is that, for the purpose of path computation, link weights are set proportional to the number of available frequency slots, in an attempt to balance the load across the network links. On the other hand, DLBSPP adopts a restoration strategy to recover from the simultaneous failure of multiple links. In the event of a multi-link failure, the algorithm attempts to restore all connections for which both the working and backup paths become unavailable. To this end, affected connections are prioritized in decreasing order of their spectrum demands, and the algorithm computes, on the fly, a new path for each connection, starting with the connection with the highest priority (i.e., the highest data rate). The motivation for this approach, referred to as traffic adaptive restoration (TAR), is based on the observation that it is more difficult to find paths for high-rate connections, hence these should be accommodated before allocating resources to low-rate connections.

A pure restoration scheme for EONs was presented in [42] as part of a study to evaluate the benefits of elastic bandwidth allocation relative to fixed-grid optical networks. Initially, only a working path is provisioned for each connection; however, if a single working path that can support the requested data rate is not available, at most two working paths of lower data rates may be assigned to a connection. Upon occurrence of a link failure, new routes are dynamically computed for all connections affected by the failure. Furthermore, if there are multiple connections with the same source and destination with failed working paths, they are all aggregated into a single connection so as to minimize the amount of resources required for restoration.

Recall that restoration schemes do not reserve backup resources for each connection. A key challenge, therefore, is to provision backup resources to ensure that all connections affected by a single failure can be fully recovered. This problem is referred to as the spare capacity allocation (SCA) problem, and it has been addressed in the context of EONs in [76]. This study considered span restoration, a failure-dependent restoration scheme whereby all connections affected by a span failure are redirected over a different path that connects the two endpoints of the failed span. A MILP formulation of the SCA problem was presented in [76] under the assumption that each node supports *spectrum conversion*; hence, the formulation did not include the continuity constraints that were included in variants of the RSA problem that we discussed earlier. The MILP uses a path-based formulation with the dual objective of minimizing spare capacity and maximizing restorable traffic capacity. The study showed that both working and spare capacity increase linearly with the traffic load, but that spare capacity redundancy improves (i.e., drops) due to more opportunities for spare capacity sharing.

2.7 Multi-Path RSA

Multi-path routing is a technique widely used in SONET networks to improve both the network-wide utilization of link capacity and the survivability of individual connections [77--79]. Multi-path routing introduces flexibility in the design and operation of the network by allowing operators to split traffic demands into multiple streams that are routed independently of each other to the destination. It also affords more options in recovering from network failures. These features can be useful in addressing and overcoming the fragmentation of spectal resources in an elastic optical network.

An algorithm that uses a hybrid single-/multi-path routing (HSMR) scheme to provision a request was presented in [48]. The algorithm considers a set of candidate paths in order of increasing weight, and takes into account fragmentation as follows. For each path, it allocates bandwidth to the request based on the number of contiguous slots (if any) available on the path and the modulation level (which depends on path distance). If the allocated bandwidth is not sufficient for the requested data rate, the algorithm proceeds to allocate bandwidth on the next path, and so on, until a sufficient number of slots has been assigned; the request is blocked if the sum of the available slots on the candidate paths is not sufficient to support its data rate. This technique overcomes the limitations of fragmentation by using multi-path routing whenever the request cannot be provisioned along a single path. Two versions of HSMR were studied, a static one in which paths are computed in advance and an adapative one that computes paths at the time a request arrives. In the latter case, the link weights were calculated using a metric that captures their fragmentation status. Similar single-/multi-path routing schemes are presented in [80]. The two algorithms in [80] differ in the way they compute paths. The first algorithm computes the paths at the time the request arrives, and in doing so, it takes into account the availability of sufficient bandwidth. The second algorithm uses pre-computed paths, and considers them in order of increasing delay when assigning bandwidth to a request. A main difference between the algorithms in [80] and the ones in [48] is that the former take the differential delay into account when allocating paths.

The dynamic multi-path service scheme presented in [81] selects the paths on which to provision a request such that the maximum differential delay along these paths does not exceed a given threshold. The algorithm also imposes a minimum bandwidth allocation granularity on each path, so as to avoid splitting the service across a large number of routing paths.

An offline version of the multi-path RSA problem was studied in [82] in the context of spporting high-speed Ethernet in EONs. The problem of establishing a set of connections, each potentially using more than one path, is formulated as a path-based ILP; in addition to the usual spectrum continuity and contiguity constraints, this ILP also imposes differential delay constraints on the multiple paths for a given connection. An efficient two-step heuristic is also presented. The heuristic first computes a set of fiber-level paths for each source-destination pair. The second phase takes these paths as input, and first attempts to find a single path for each request. If a single path is not found, then the algorithm proceeds to compute a parallel transmission solution across multiple paths.

Multi-path solutions in EONs have also been investigated in the context of protection and restoration. The single-path provisioning, multi-path recovery (SPP-MPR) scheme in [83] provisions a demand on a single primary path. However, it provides for a multi-path squeezed recovery mechanism that utilizes two backup paths and allows for bandwidth squeezing similar to [73, 74]. The problem is formulated as an MILP, but a genetic algorithm-based meta-heuristic is also presented.

An offline multi-path provisioning scheme for OFDM-based EONs was first introduced in [84]. The problem, referred to as the "static survivable multi-path routing and spectrum allocation" (SM-RSA) was first formulated as a path-based ILP. A three-step heuristic was also developed. In the first step, Bhandari's algorithm [85] is used to compute the largest number of disjoint paths between each source-destination pair. In the second step, requests are considered either in order of decreasing demands or decreasing path length, and are assigned paths and spectrum blocks. In the third step, requests on paths that use the maximum number of spectral slots are rerouted in an attempt to reduce the use of spectral resources. An online version of the same problem is studied in [86] and again an ILP formulation and a heuristic algorithm are presented. Both the ILP and heuristic limit the number of paths allocated to each connection to at most three; they also allow for connections that require partial (not full) protection.

Chapter 3

A Multiprocessor Scheduling Perspective for SA & RSA in EONs

As we discussed in Chapter 2, although there exist different versions of ILP formulations such as link-based, path-based, and channel-based for the RSA problem, these formulations have shortcomings in solving practical network topologies' instances. Hence, heuristic algorithms are developed to tackle the RSA problem. In this chapter, we provide a new insight into the SA problem in mesh networks in Section 3.1 by showing that this problem can be transformed to the scheduling multiprocessor tasks on dedicated processors denoted as $P|fix_j|C_{max}$. Similarly, we show in Section 3.2 that the RSA problem with fixed-alternate routing in general-topology (mesh) networks is a special case of multiprocessor scheduling problem $P|set_j|C_{max}$.

3.1 The SA Problem in General Topology Networks

Considering the definition 2.1.2 of the SA in Chapter 2, we now show that the SA problem can be viewed as a problem of scheduling tasks on multiprocessor systems in which tasks may require more than one processor simultaneously. Consider the following scheduling problem that has been studied extensively in the literature [87, 88]:

Definition 3.1.1 **($P|fix_j|C_{max}$)** *Given*

- *a set of m identical processors,*
- *a set of n tasks with processing time $p_j, j = 1, \ldots, n$, and*
- *a prespecified set fix_j of processors for executing each task $j, j = 1, \ldots, n$,*

schedule the tasks so as to minimize the makespan $C_{max} = \max_j C_j$, where C_j denotes the completion time of task j, under the constraints:

1. *preemptions are not allowed;*

2. *each task must be processed simultaneously by all processors in set fix_j; and*

3. *each processor can work on at most one task at a time.*

We denote by $Pm|fix_j|C_{max}$ the special case of $P|fix_j|C_{max}$ in which the number of processors m is considered to be fixed. It has been shown [88] that the three-processor problem $P3|fix_j|C_{max}$ is strongly NP-hard for general processing times, but that if the number of processors m is fixed and all tasks have unit times, i.e., $Pm|fix_j, p_j = 1|C_{max}$, then the problem is solvable in polynomial time. Approximation algorithms and/or polynomial time approximation schemes (PTAS) have been developed for several versions of the problem [89].

The next two lemmas show that the SA problem in networks of general topology is a special case of the $P|fix_j|C_{max}$ scheduling problem, but the reverse is not true.

Lemma 3.1.1 *SA on general (mesh) networks transforms to $P|fix_j|C_{max}$.*

Proof. Consider an instance of the SA problem on a network of general topology represented by graph $G = (\mathcal{V}, \mathcal{A})$, demand matrix $T = [t_{sd}]$, and path set $\{r_{sd}\}$. Construct an instance of $P|fix_j|C_{max}$ as follows. For each arc $a_k \in \mathcal{A}, k = 1, \cdots, |\mathcal{A}|$, there is a processor k. For each spectrum demand t_{sd}, there is a task j with $p_j = t_{sd}$ and $fix_j = \{k : a_k \in r_{sd}\}$. In other words, the amount of spectrum of a demand transforms to the processing time of the corresponding task, and the links of its path to the processors that the task requires. Due to the non-overlapping spectrum constraint, each processor may work on at most one task at a time, due to the spectrum continuity constraint, each task must be processed simultaneously by all its processors, whereas due to the spectrum contiguity constraint, preemptions are not allowed. By construction, the amount of spectrum assigned to any arc of G in a solution of the SA instance is equal to the completion time of the last task scheduled on the corresponding processor, hence minimizing the spectrum on any link in the SA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|fix_j|C_{max}$. ■

The above lemma shows that any instance of the SA problem can be transformed into an instance of the $P|fix_j|C_{max}$ problem, and hence, an algorithm for solving the latter problem may be used for solving the former one. However, the reverse of Lemma 3.1.1 is not true. In other words, there exist instances of $P|fix_j|C_{max}$ for which there is no corresponding instance of the SA problem, as we now show.

Lemma 3.1.2 *There exist instances of $P|fix_j|C_{max}$ for which there is no corresponding instance of the SA problem.*

Proof. By counterexample. Consider an instance of $P4|fix_j|C_{max}$ with these five tasks whose processing times can be arbitrary:

task	fix_j
τ_1	$\{1,2\}$
τ_2	$\{2,3\}$
τ_3	$\{3,4\}$
τ_4	$\{4,1\}$
τ_5	$\{2,4\}$

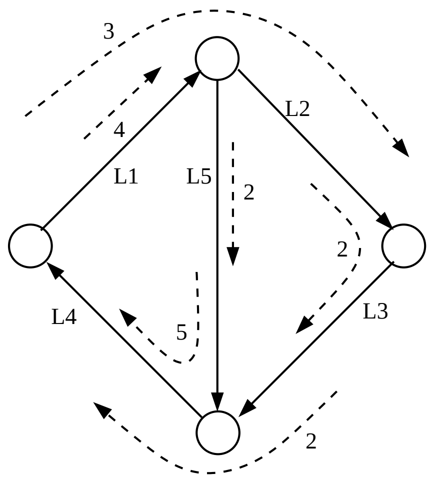
Because of the first four tasks, the graph of the corresponding SA instance would have to be the four-link unidirectional ring network such that link 1 is adjacent to 2, 2 is adjacent to 3, 3 to 4, and 4 to 1. But then, there is no path r_{sd} for the spectrum demand corresponding to the last task, hence an instance of SA does not exist. ■

In conclusion, Lemma 3.1.1 and Lemma 3.1.2 state the SA problem is a subset of the $P|fix_j|C_{max}$ scheduling problem. As an example, Figure 3.1(a) shows an instance of the SA problem on a mesh network with five directed links, $L1, L2, L3, L4$, and $L5$. There are five demands, shown as dotted lines, with the number of slots required by each demand shown next to the corresponding line. Figure 3.1(b) shows the optimal schedule for the $P|fix_j|C_{max}$ problem corresponding to this SA instance, whereby link $L1$ maps to processor $P1$, link $L2$ to processor $P2$, and so on. As we can see, the demand of size 3 that follows the path $L1$-$L2$ is mapped to a task that is scheduled in the time interval $[4,7]$ on the corresponding processors $P1$ and $P2$; similarly for the other demands. The schedule is optimal in that $C_{max} = 7$ is equal to the total processing time required for processors $P1, P4$ and $P5$. Also, the value of C_{max} is equal to the total number of spectrum slots required for links $L1, L4$, and $L5$.

The following lemma shows that the SA problem in unidirectional rings with as few as three links is NP-hard.

Lemma 3.1.3 *The SA problem in unidirectional rings is NP-hard.*

Proof. The $P3|fix_j|C_{max}$ problem can be transformed to the SA problem on a unidirectional three-link ring, where each processor corresponds to a link, and each task j corresponds to the traffic demand on the segment of the ring defined by the links in fix_j. Since $P3|fix_j|C_{max}$ is NP-complete [88], the same is true for the SA problem on the three-link ring. ■



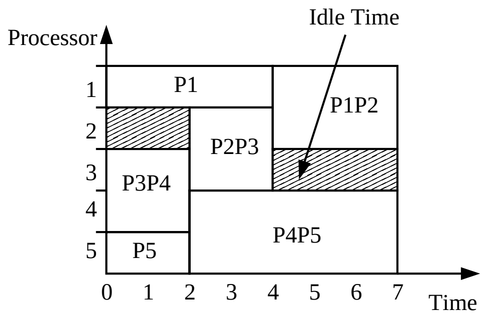


Figure 3.1: (a) Instance of the SA problem on a mesh network with five directed links (arcs). (b) Optimal schedule of the corresponding $P|fix_j|C_{max}$ problem

3.2 The RSA Problem in General Topology Networks

Consider the general definition 2.1.1 of the RSA problem with fixed-alternate routing in elastic optical networks. We now show that the RSA problem with fixed-alternate routing in networks of general topology can also be viewed as $P|set_j|C_{max}$ problem. Consider the following scheduling problem that has been studied in the literature [90--92]:

Definition 3.2.1 **($P|set_j|C_{max}$)** *Given*

- *a set of m identical processors,*
- *a set of n tasks with processing time*
- *and a prespecified set $set_j = \{fix_j^1, \ldots, fix_j^k\}$ of k alternative processor sets to which task j can be assigned,*

schedule the tasks so as to minimize the makespan $C_{max} = \max_j C_j$, where C_j denotes the completion time of task j, under the constraints:

1. *preemptions are not allowed,*
2. *each task must be processed simultaneously by all processors in only one of the processor sets in set_j, and*
3. *each processor can work on at most one task at a time.*

Clearly, $P|fix_j|C_{max}$ is a special case of $P|set_j|C_{max}$ with $k = 1$ processor set for each task. It has been shown that, in the general case, there can be no constant-ratio polynomial time approximation algorithm for $P|set_j|C_{max}$ unless $P = NP$ [93]. The two-processor problem $P2|set_j|Cmax$ has been proven in [94] to be NP-hard. Approximation algorithms with ratio m and $m/2$ have been developed in [90] and [91], respectively, for problem $Pm|set_j|C_{max}$, in which the number m of processors is considered to be fixed and is not part of the input (as it is for $P|set_j|C_{max}$). Also, polynomial time approximation schemes (PTAS) for $Pm|set_j|C_{max}$ were introduced in [92, 95].

The following two lemmas show that the RSA problem with fixed-alternate routing in mesh networks is a special case of $P|set_j|C_{max}$.

Lemma 3.2.1 *RSA with fixed-alternate routing in mesh networks transforms to* $P|set_j|C_{max}$.

Proof. Consider an instance of RSA with fixed-alternate routing on a network with a general topology graph $G = (\mathcal{V}, \mathcal{A})$, demand matrix $T = [t_{sd}]$ and set $\{r_{sd}^1, \dots, r_{sd}^k\}$ of alternate routes for source-destination pair (s, d). Construct an instance of $P|set_j|C_{max}$ such that:

- for each directed arc $a_l \in \mathcal{A}$, there is a processor l, and
- for each spectrum demand t_{sd}, there is a task j with $p_j = t_{sd}$ and $set_j = \{fix_j^1, \dots, fix_j^k\}$, where $fix_j^i = \{q : a_q \in \{r_{sd}^i\}\}$.

Hence, the amount of spectrum of a demand transforms to the processing time of the corresponding task, the set of alternate paths to the set of alternate processor sets for that task, and the links of each alternate path to the corresponding set of alternate processors for the task. Due to the spectrum contiguity constraint, preemptions are not allowed. The spectrum continuity constraint guarantees that each task will be processed simultaneously by all processors in the alternate set assigned to the task, whereas the non-overlapping spectrum constraint requires that each processor work on at most one task at a time.

By construction, the amount of spectrum assigned to any arc of G in a solution of the RSA instance is equal to the completion time of the last task scheduled on the corresponding processor, hence minimizing the spectrum on any link in the RSA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|set_j|C_{max}$. ■

We now show that the reverse of Lemma 3.2.1 is not true. In other words, there exist instances of $P|set_j|C_{max}$ for which there is no corresponding instance of the RSA problem.

Lemma 3.2.2 *There exist instances of* $P|set_j|C_{max}$ *for which there is no corresponding instance of the RSA problem.*

Proof. We prove this statement by providing a counterexample. Consider an instance of $P|set_j|C_{max}$ with eight processors $\{1, 2, 3, 4, 1', 2', 3', 4'\}$, and these five tasks whose processing times can be arbitrary:

task	fix_j^1	fix_j^2
τ_1	$\{1, 2\}$	$\{4', 3'\}$
τ_2	$\{2, 3\}$	$\{1', 4'\}$
τ_3	$\{3, 4\}$	$\{2', 1'\}$
τ_4	$\{4, 1\}$	$\{3', 2'\}$
τ_5	$\{2, 4\}$	$\{4', 2'\}$

Because of the first four tasks, the graph of the corresponding RSA instance would have to be the four-node, eight-link bidirectional ring network such that: (1) links l and $l', l = 1, 2, 3, 4$, are links in the clockwise and counter-clockwise direction, respectively, between adjacent nodes in the ring, (2) in the clockwise direction, link 1 is adjacent to 2, 2 is adjacent to 3, 3 to 4, and 4 to 1, and (3) similarly for links in the counter-clockwise direction. Since there are no feasible paths for the spectrum demand corresponding to the last task τ_5, an instance of RSA does not exist. ■

From our discussion in Lemma 3.2.1 and Lemma 3.2.2, we conclude that the RSA problem is a subset of the $P|set_j|C_{max}$ scheduling problem.

Chapter 4

The SA Problem in Chain EONs

We now analyze the SA problem in chain EONs in Section 4.1 using the multiprocessor scheduling perspective introduced in Chapter 3. Then, we show in Section 4.2 that the SA problem in chain (path) networks is NP-hard for four-link networks, but it is solvable in polynomial time for three-link networks. In Section 4.3, we develop new constant-ratio approximation algorithms for the SA problem in chain (path) networks in which the number of links is fixed. We also introduce a suite of list scheduling algorithms in Section 4.4. Finally, we present numerical results to demonstrate the effectiveness of the algorithms in Section 4.5.

4.1 The SA Problem in Chain Networks

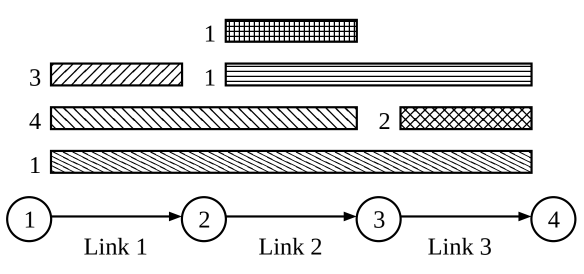
In chain (linear) networks, the route r_{sd} of each traffic demand is uniquely determined by its source and destination nodes. Let us define the following special case of problem $P|fix_j|C_{max}$:

Definition 4.1.1 **($P|line_j|C_{max}$)** *The $P|fix_j|C_{max}$ problem under the additional constraint that the identical processors are labeled $1, 2, 3, \ldots,$ and the prespecified set $line_j$ of processors for each task j consists of processors with consecutive labels.*

The following result states that the SA problem on a chain with m links is equivalent to $P|line_j|C_{max}$, hence an algorithm for solving $P|line_j|C_{max}$ may be used to solve SA, and vice versa.

Lemma 4.1.1 *The SA problem on a graph G that is a chain with m links is equivalent to $P|line_j|C_{max}$.*

Proof. First consider an instance of the SA problem on a chain G with $m+1$ nodes labeled $1, 2, \ldots, m+1$, and m arcs $a_1 =<1, 2>$, $a_2 =<2, 3>$,..., a_m $<m, m+1>$, and spectrum demand matrix $T = [t_{sd}]$ such that $t_{sd} = 0$ if $s \geq d$. Given this SA instance, the steps of the proof



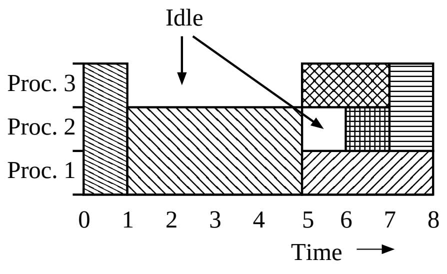


Figure 4.1: (a) An instance of the offline RSA problem on a directed path; (b) The optimal schedule for the corresponding $P3|line_j|C_{max}$ problem

of Lemma 3.1.1 will construct a valid instance of $P|line_j|C_{max}$ since the sets fix_j consist of processors with consecutive labels.

Given an instance of $P|line_j|C_{max}$, we construct an instance of the SA problem on a chain graph G as follows. The graph has $m+1$ nodes labeled $1, 2, \ldots, m+1$, and m arcs, such that for each processor $k, k = 1, \ldots, m$, there is an arc $a_k = <k, k+1>$. For each task j with $line_j = \{s, \ldots, d-1\}$, there is a traffic demand with $t_{sd} = p_j$ and route $r_{sd} = \{<s, s+1>, \ldots, <d-1, d>\}$. It is not difficult to verify that the three constraints of $P|line_j|C_{max}$ ensure that the three constraints of SA are satisfied, and that minimizing the makespan C_{max} minimizes the maximum amount of spectrum on any arc of G. ■

In theory, algorithms developed for $P|fix_j|C_{max}$ may be applied to schedule instances of $P|line_j|C_{max}$. Due its difficulty, however, there are limited results for the former problem, and even algorithms for a small number of processors (e.g., $m = 3, 4, 5$) can be quite complex and not practical for a network operator.

In the following section, we determine the complexity of $Pm|line_j|C_{max}$. Following that, we present approximation algorithms for the most general version of the problem, i.e., for any number of processors and any task sizes.

4.2 Complexity Results for $Pm|line_j|C_{max}$

We first show that there is a polynomial time algorithm for $P3|line_j|C_{max}$. Recall that problem $P3|fix_j|C_{max}$ is strongly NP-hard [88], hence the additional constraint that the set of processors in $line_j$ have consecutive labels makes the problem tractable for three processors. Furthermore, note that whereas it was stated in [20] that the SA problem in chains is NP-hard, this result implies that the problem is in fact polynomial on three-link chains.

Lemma 4.2.1 *$P3|line_j|C_{max}$ may be solved in polynomial time.*

Proof. The proof is by construction of the optimal schedule. Tasks that require all three processors cannot be executed in parallel with any other task, and hence they may be simply added at the beginning or end of the schedule without affecting optimality. Therefore, we focus our attention on tasks requiring either one or two processors, i.e., tasks with $line_j = \{1\}, \{2\}, \{3\}, \{1,2\}$, or $\{2,3\}$. Without loss of generality, let processor 1 be the dominant processor, i.e., the one that requires the most processing time; the case where either processor 2 or processor 3 are dominant can be handled in a very similar manner. Construct the following schedule. Tasks with $line_j = \{1,2\}$ are executed back-to-back without any idle time, followed by tasks with $line_j = \{1\}$. Let t be the time when the last task with $line_j = \{1\}$ completes. Schedule tasks with $line_j = \{2,3\}$, without any idle time between them, at the end of the schedule and in parallel with tasks with $\{line_j = 1\}$, so that the last one finishes at time t. Then, schedule tasks with $line_j = \{2\}$ and $line_j = \{3\}$ before tasks with $line_j = \{2,3\}$. Clearly, these tasks can fit in the schedule since processors 2 and 3 are not dominant. The schedule is optimal since the dominant processor is never idle, and hence the makespan $C_{max} = t$, the time required to execute the tasks on the dominant processor. ■

As an example, consider the RSA problem instance on a chain network with 4 nodes and 3 links, as shown in Figure 4.1(a). For this instance, the spectrum demand matrix is:

$$T = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The corresponding $P3|line_j|C_{max}$ instance has six tasks:

task	p_j	$line_j$
τ_1	1	$\{1,2,3\}$
τ_2	4	$\{1,2\}$
τ_3	3	$\{1\}$
τ_4	1	$\{2,3\}$
τ_5	1	$\{2\}$
τ_6	2	$\{3\}$

The optimal schedule constructed as described in the proof of Lemma 4.2.1 is shown in Figure 4.1(b). In this schedule, tasks requiring all three processors are executed first.

The following theorem shows that the problem $Pm|line_j|C_{max}$ is NP-complete for four or more processors. The proof is based on a reduction from the PARTITION problem [96] which is

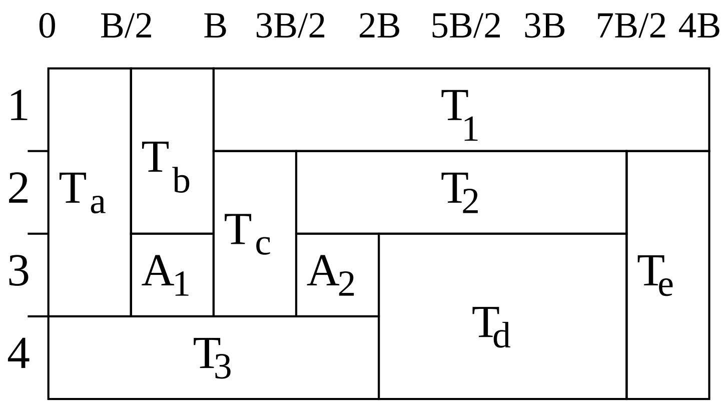


Figure 4.2: Feasible schedule with $C_{max} = 4B$

defined as:

Definition 4.2.1 (PARTITION) *Given a set of k integers $A = \{a_1, a_2, \dots, a_k\}$ such that $B = \sum_{j=1}^{k} a_j$, does there exist a partition of A into two sets, A_1 and A_2, such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = \frac{B}{2}$?*

Theorem 4.2.1 *$P4|line_j|C_{max}$ is NP-complete.*

Proof. Given an instance of PARTITION, we create an instance of $P4|line_j|C_{max}$ as follows. For each $a_j \in A$ we create a task τ_j with processing time $p_j = a_j$ and $line_j = \{3\}$. Moreover, we create the following gadget tasks:

task	p_j	$line_j$
T_a	$B/2$	$\{1,2,3\}$
T_b	$B/2$	$\{1,2\}$
T_c	$B/2$	$\{2,3\}$
T_d	$3B/2$	$\{3,4\}$
T_e	$B/2$	$\{2,3,4\}$
T_1	$3B$	$\{1\}$
T_2	$2B$	$\{2\}$
T_3	$2B$	$\{4\}$

If there is a partition of A into A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then we can schedule the jobs as shown in Figure 4.2 and get a feasible schedule with $C_{max} = 4B$.

Let us now assume that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 4B$. Without loss of generality, assume that T_a is executed before T_e in $\mathcal{S}$; otherwise, we can use similar arguments and reach the same conclusion. Then, it must be that T_3 is executed before both T_d and T_e. Moreover, T_d must be executed before T_e, otherwise it would not be possible to schedule task T_2 for the schedule to have length at most $4B$. Hence, tasks T_a, T_3, T_d, and T_e must be scheduled in the order shown in Figure 4.2. Moreover, tasks T_a and T_b must be scheduled before T_1, while task T_c must be scheduled before T_2 for the schedule length to be at most $4B$. If T_a is scheduled before T_b (as shown in Figure 4.2), then on processor 3, only the intervals $[B/2, B]$ and $[3B/2, 2B]$ are available for the execution of the PARTITION jobs. If T_b is scheduled before T_a, then on processor 3, only the intervals $[0, B/2]$ and $[3B/2, 2B]$ are available for the PARTITION jobs. In both cases, a partition exists. ■

As stated in [20], the above result confirms that the SA problem on chains with four or more links is harder than the wavelength assignment problem which can be solved in polynomial time on chains.

4.3 Approximation Algorithms for $P|line_j|C_{max}$

We first show that there exist 1.5-approximation algorithms for four and five processors.

Lemma 4.3.1 *There exists a 1.5-approximation algorithm for $P4|line_j|C_{max}$.*

Proof. The proof is by construction and discussed in Appendix A. ■

Lemma 4.3.2 *There exists a 1.5-approximation algorithm for $P5|line_j|C_{max}$.*

Proof. The proof is by construction and discussed in Appendix B. ■

4.3.1 Two-Stage Approximation Algorithms

We now introduce a two-stage algorithm for $P|line_j|C_{max}$, and show that it yields a constant approximation ratio for any number of processors $m \geq 6$. The algorithm, described in Figure 4.3, considers three sets of processors based on their labels: a set consisting of the $k < m$ processors with low index labels $1, \ldots, k$, a set of l processors, $l + k < m$, with labels $k + 1, \ldots, k + l$, and a set of $m - k - l$ processors with the high index labels $k + l + 1, \ldots, m$. We partition the set of tasks into three sets:

- set $\mathcal{T}_{mid}$ consists of tasks that require at least one of the l middle processors (and may also require processors from one or both of the other sets);

- set $\mathcal{T}_{lo}$ contains tasks requiring only processors in the set of k low index processors (and no other processor); and
- set $\mathcal{T}_{hi}$ is composed of tasks that require only processors in the set of $m-k-l$ high index processors (and no other processor).

The key idea is based on the observation that the set of tasks $\mathcal{T}_{lo}$ may be scheduled in parallel with the set of tasks $\mathcal{T}_{hi}$. Therefore, we use an optimal or approximation algorithm to schedule the tasks in each set separately, creating three schedules, $\mathcal{S}_{mid}$, $\mathcal{S}_{lo}$, and $\mathcal{S}_{hi}$, respectively. The final schedule for the original problem consists of two stages: in the first stage, schedule $\mathcal{S}_{mid}$ is executed individually, while in the second stage, schedules $\mathcal{S}_{lo}$ and $\mathcal{S}_{hi}$ are executed in parallel.

We have the following result.

Lemma 4.3.3 *Let α_{mid}, α_{lo}, and α_{hi} be the approximation ratio of the algorithms used to schedule tasks in sets $\mathcal{T}_{mid}$, $\mathcal{T}_{lo}$, and $\mathcal{T}_{hi}$, respectively (the approximation ratio is 1 if an optimal algorithm exists). Then the two-stage algorithm of Figure 4.3 is an approximation algorithm for the original problem with ratio:*

$$\alpha \;=\; \alpha_{mid} \;+\; \max\{\alpha_{lo}, \alpha_{hi}\}. \tag{4.1}$$

Proof: The proof follows from the fact that the makespan of an optimal schedule for each of the three sets of tasks is no longer than the makespan of an optimal schedule for the original set of tasks. ■

Figure 4.4 shows a two-stage schedule for $m = 9$ processors in which $k = l = m - k - l = 3$. Due to Lemma 4.2.1, the $P3|line_j|C_{max}$ problem can be solved optimally in polynomial time, hence $\alpha_{mid} = \alpha_{lo} = \alpha_{hi} = 1$. Therefore, the two-stage algorithm is a 2-approximation algorithm for $m = 9$ processors (and also for problems with $m = 6, 7, 8$ processors).

For problems with $m = 10 - 13$ processors, we consider the middle three processors to obtain task set $\mathcal{T}_{mid}$, and apply the 1.5-approximation algorithm for four or five processors to schedule the other two tasks sets, resulting in an approximation ratio of 2.5. For problems with $m = 14, 15$ processors, we obtain an approximation ratio of 3 by considering tasks sets on four or five processors. Finally, we note that for $m > 15$, we may apply the two-stage algorithm recursively to schedule each set of tasks. For instance, for $m = 19$, we can schedule the tasks that require at least one of the middle nine processors with a makespan that is no more than twice the optimal, as in Figure 4.4. Applying the 1.5-approximation algorithm to schedule the tasks requiring at least one of the five lower (respectively, higher) index processors, the overall approximation ratio of the two-stage algorithm is 3.5.

```
Two-Stage Approximation Algorithm for $P|line_j|C_{max}$, $m \geq 6$
Input: A set $\mathcal{T}$ of $n$ tasks on $m$ processors, each task $j$ having a processing time $p_j$ and a set $line_j \subseteq \{1, 2, \ldots, m\}$ of required processors
Output: A two-stage schedule of tasks

begin
        // Sets of low, mid, and high index processors
    1.  $lo \leftarrow \{1, \ldots, k\}$    // $k < m$
    2.  $mid \leftarrow \{k+1, \ldots, k+l\}$    // $k + l < m$
    3.  $hi \leftarrow \{k+l+1, \ldots, m\}$
        // Set of tasks $\mathcal{T}$ for each subproblem
    4.  $\mathcal{T}_{mid} \leftarrow \{j \in \mathcal{T} : \{k+1, \ldots, k+l\} \cap line_j \neq \emptyset\}$
    5.  $\mathcal{T}_{lo} \leftarrow \{j \in \mathcal{T} \setminus \mathcal{T}_{mid} : \{1, \ldots, k\} \cap line_j \neq \emptyset\}$
    6.  $\mathcal{T}_{hi} \leftarrow \{j \in \mathcal{T} \setminus \mathcal{T}_{lo} \setminus \mathcal{T}_{mid}\}$
    7.  Schedule tasks in $\mathcal{T}_{mid}$ using an optimal or approximation
        algorithm for the corresponding $Pl|line_j|C_{max}$ problem
    8.  Schedule tasks in $\mathcal{T}_{lo}$ using an optimal or approximation
        algorithm for the corresponding $Pk|line_j|C_{max}$ problem
    9.  Schedule tasks in $\mathcal{T}_{hi}$ using an optimal or approximation
        algorithm for the corresponding
        $P(m-k-l)|line_j|C_{max}$ problem
    10. return a schedule of two stages:
        the first stage consists of the schedule for tasks $\mathcal{T}_{mid}$
        the second stage consists of the schedule for tasks $\mathcal{T}_{lo}$
        and $\mathcal{T}_{hi}$ executed in parallel
end
```

Figure 4.3: A two-stage scheduling algorithm for $P|line_j|C_{max}$

More generally, the approximation ratio $\alpha(m)$ for any number m of processors can be computed using the recurrence relationship:

$$\alpha(m) = \begin{cases} 1 & m = 1, 2, 3 \\ 1.5 & m = 4, 5 \\ \min\limits_{l=1,\cdots,m-2} \left\{\alpha(l) + \alpha\left(\lceil \frac{m-l}{2} \rceil\right)\right\} & m \geq 6 \end{cases} \tag{4.2}$$

The recursive expression can be explained by making two observations. First, for a given number l of processors in the set $\mathcal{T}_{mid}$, the approximation ratio in (4.1) is minimized when the number of processors in the sets $\mathcal{T}_{lo}$ and $\mathcal{T}_{hi}$ differ by at most one; in this case, the number of processors in the larger of the two sets is $\lceil \frac{m-l}{2} \rceil$), and determines the result of the maximum operation in expression (4.1). Second, all three sets of processors must be non-empty, hence the range of values of l in the expression. Therefore, the approximation ratio for a given value of l is $\alpha(l) + \alpha(\lceil \frac{m-l}{2} \rceil)$ and the best ratio can be obtained by taking the minimum over all possible

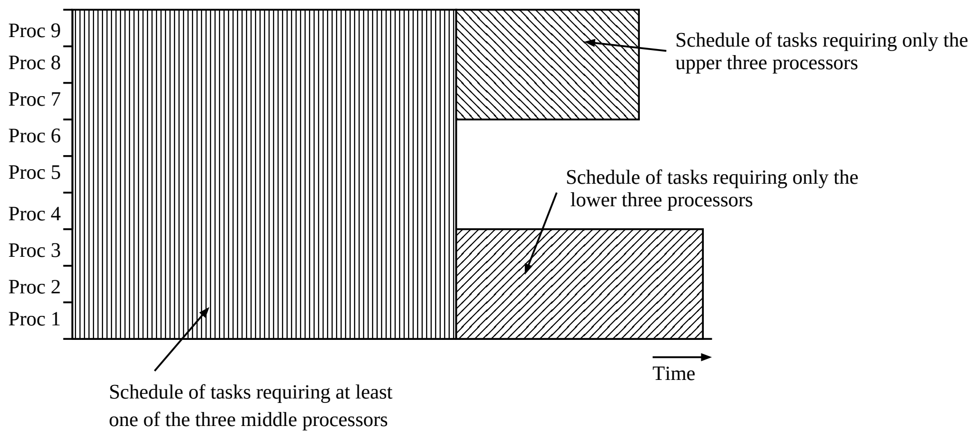

Proc 9
Proc 8
Proc 7
Proc 6
Proc 5
Proc 4
Proc 3
Proc 2
Proc 1
Schedule of tasks requiring only the upper three processors
Schedule of tasks requiring only the lower three processors
Schedule of tasks requiring at least one of the three middle processors
Time

Figure 4.4: 2-approximation schedule for $m = 9$ processors

values of l.

4.4 List Scheduling Algorithms for $Pm|line_j|C_{max}$

In this section, we present a suite of list scheduling algorithms for solving the $Pm|line_j|C_{max}$ problem; due to Lemma 4.1.1, these algorithms may also be used to solve the SA problem on chains. The algorithms attempt to minimize the makespan C_{max} by identifying *compatible* tasks, i.e., sets of tasks that may be executed simultaneously on the multi-processor system. More formally, we have the following definition.

Definition 4.4.1 *A set $\mathcal{T}$ of tasks for the $Pm|line_j|C_{max}$ problem are said to be compatible if and only if their prespecified sets of processors are pairwise disjoint, i.e., $line_j \cap line_i = \emptyset, \forall\ i, j \in \mathcal{T}$.*

We present two broad classes of algorithms that differ in the granularity at which they make scheduling decisions. The first class of algorithms assemble compatible tasks into *blocks*, and schedule a whole block of tasks at a time. The second class of algorithms operate at finer granularity and make scheduling decisions at the level of individual tasks and at finer time scales.

4.4.1 Block-Based Scheduling Algorithms

These algorithms proceed by constructing *blocks* of compatible tasks. Specifically, blocks of compatible tasks are scheduled such that:

- all tasks in a block begin execution at the same time t, and

- there is no idle time between the completion of the longest task in a block and the beginning of the next block.

The input to the algorithm is a list of tasks. The algorithm assembles a block by selecting the first task in the list, and then scanning the remaining tasks (in the order listed) to identify tasks compatible with the ones already in the block. A block is considered complete if no additional compatible tasks exist; the algorithm removes all the tasks of the block from the list and continues to build the next block, until all tasks have been scheduled.

A pseudocode description of a block-based scheduling algorithm is presented in Figure 4.5. The running time complexity of the algorithm is determined by the two **while** loops. In the worst case, each loop is executed at most n times, where n is the number of tasks. Hence, the overall running time of the algorithm is $O(n^2)$. This overall complexity is not affected by accounting for the time it takes to create the list of tasks given as input to the algorithm, as creating this list would typically involve sorting the n tasks according to some attribute.

We identify two block-based scheduling algorithms that differ in the order in which the tasks are sorted in the initial list of tasks passed to the algorithm.

- **Longest First Block-based Algorithm (LFB).** Tasks are sorted in decreasing order of their processing times p_j.
- **Widest First Block-based Algorithm (WFB).** Tasks are sorted in decreasing order of the number $|line_j|$ of processors they require.

4.4.2 Compact Scheduling Algorithms

Block-based schedules are simple to create and implement in that each task in a block starts execution at exactly the same time. However, the fact that tasks within a block have varying processing times may result in long idle times for some processors. Consequently, the makespan of the final schedule may be longer than necessary. We now present a class of scheduling algorithms that attempt to minimize the makespan by eliminating or reducing such idle times. Rather than assembling blocks of tasks and making scheduling decisions at the end of each block, these algorithms select individual tasks for execution at finer scheduling instants resulting in more compact schedules. The scheduling instants consist of:

- the start time of the schedule (i.e., $t = 0$), and
- the instant each task completes execution.

The input to a compact algorithm is a list of tasks. The algorithm maintains a list of idle processors. At each scheduling instant t, the algorithm scans the list to identify tasks that are

Block-Based Scheduling Algorithm for $Pm|line_j|C_{max}$
Input: A list L of n tasks on m processors, each task j having a processing time p_j and a set $line_j \subseteq \{1, 2, \ldots, m\}$ of required processors
Output: A schedule of task blocks, each block consisting of compatible tasks, such that each block starts execution immediately after the previous block completes

```
begin
    1. $b \leftarrow 1$    //$b$ is the block index
    2. $S_b \leftarrow 0$   //The time that block $b$ starts execution
    3. $P_b \leftarrow \emptyset$  //The union of sets $line_j$ of tasks in block $b$
    4. while list $L \neq \emptyset$ do
    5.     Remove the first task $j$ from list $L$
    6.     Add task $j$ to block $b$
    7.     $P_b \leftarrow P_b \cup line_j$
    8.     while not at end of list $L$ or $P_b \neq \{1, 2, \ldots, m\}$ do
    9.         $k \leftarrow$ first task in list
    10.        if $line_k \cap P_b = \emptyset$ then
    11.            Remove the task $k$ from list $L$
    12.            Add task $k$ to block $b$
    13.            $P_b \leftarrow P_b \cup line_k$
    14.    end while  // no more tasks may be added to block $b$
    15.    $b \leftarrow b + 1$
    16.    $S_b \leftarrow S_{b-1} + \max_{j \in b-1} p_j$
    16.    $P_b \leftarrow \emptyset$
    17. end while
    18. return tasks in each block, block starting times $S_b$
end
```

Figure 4.5: A block-based scheduling algorithm for $Pm|line_j|C_{max}$

compatible with the currently executing ones, i.e., tasks with a set $line_j$ that is a subset of the free processors. When such a task is identified, the algorithm removes it from the list, updates the set of free processors, and continues scanning the list until no other compatible task is found. It then advances to the next scheduling instant and repeats the process while the list is not empty.

Figure 4.6 presents a pseudocode description of a compact scheduling algorithm. The running time complexity of the algorithm is $O(n^2)$. Similar to block-based algorithms, we distinguish two algorithms based on the order in which tasks appear in the list.

- **Longest First Compact Algorithm (LFC)**. Tasks are sorted in decreasing order of their processing times p_j.
- **Widest First Compact Algorithm (WFC)**. Tasks are sorted in decreasing order of the number $|line_j|$ of processors they require.

Compact Scheduling Algorithm for $Pm|line_j|C_{max}$
Input: A list L of n tasks on m processors, each task j having a processing time p_j and a set $line_j \subseteq \{1, 2, \ldots, m\}$ of required processors
Output: A schedule of tasks, i.e., the time S_j when each task j starts execution on the multi-processor system

```
begin
  1. $t \leftarrow 0$   //Scheduling instant
  2. $F \leftarrow \{1, \ldots, m\}$  //Set of currently idle processors
  3. while list $L \neq \emptyset$ do
  4.     $j \leftarrow$ first task in list $L$
  5.     if $line_j \subseteq F$ then
  6.         Remove the task $j$ from list $L$
  6.         $S_j \leftarrow t$ // Task $j$ starts execution at time $t$
  7.         $F \leftarrow F \setminus line_j$
  8.     while not at the end of list $L$ or $F \neq \emptyset$ do
  9.         $k \leftarrow$ first task in list
 10.         if $line_k \subseteq F$ then
 11.             Remove the task $k$ from list $L$
 12.             $S_k \leftarrow t$ Task $k$ starts execution at time $t$
 13.             $F \leftarrow F \setminus line_k$
 14.     end while  // no more tasks may start at time $t$
 15.     $j \leftarrow$ the first task executing at time $t$ to complete
 16.     $t \leftarrow S_j + p_j$
 16.     $F \leftarrow F \cup line_j$
 17. end while
 18. return the task start times $S_j$
end
```

Figure 4.6: A compact scheduling algorithm for $Pm|line_j|C_{max}$

Since compact scheduling algorithms make decisions at finer time scales, we expect that they perform better than block-based ones.

4.5 Numerical Results

In the following two subsections, we discuss two sets of experiments we carried out to evaluate the performance of the list scheduling algorithms.

4.5.1 SA in Chain Networks

In the first set of experiments, we consider instances of the $Pm|line_j|C_{max}$ problem with a relatively small number of processors, namely, $m = 5, 10, 15, 20$; such instances correspond to instances of the SA problem on chains of length typical of a commercial wide area network.

For each problem instance, we generated traffic demands between every source and destination node on the chain. The size of the traffic demands (i.e., task times) were generated using three different distributions:

- *Discrete uniform:* traffic demands may take any of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ with equal probability; these values correspond to data rates (in Gbps) to be supported by EONs.
- *Discrete high:* traffic demands may take one of the five discrete values above with probabilities $0.10, 0.15, 0.20, 0.25$, and 0.30, respectively; in other words, higher data rates have higher probability to be selected.
- *Discrete low:* traffic demands may take one of the five discrete values above with probabilities $0.30, 0.25, 0.20, 0.15$, and 0.10, respectively, such that lower data rates have higher probability to be selected.

We considered various other probability distributions on the same set of values, but the results are similar to the ones we present below and hence are omitted.

We use a distance-adaptive spectrum allocation strategy to allocate spectrum to each trafffic demand based on its data rate (in Gbps) and the length of its path (i.e., number of links, or processors in the scheduling problem) [7, 36]. We assume a 12.5 GHz slot width, and consider two modulation formats: for paths with up to (respectively, more than) ten links we assume 16-QAM (respectively, QPSK), such that demands of size 10, 40, 100, 400, and 1000 Gbps are assigned 1, 1, 2, 8, and 20 (respectively, 1, 2, 4, 16, and 40) slots, consistent with the values used in [36, Table 1].

Figures 4.7-4.9 plot the average ratio achieved by the four list scheduling algorithms, LFB, LFC, WFB, and WFC, for each of the three traffic distributions; specifically, each data point in the figures represents the average over 30 randomly generated problem instances. The average ratio for a given algorithm is defined as the ratio of the makespan value C_{max} (i.e., maximum spectrum used on any link of the corresponding chain) obtained by the algorithm for a given problem instance over the lower bound (i.e., the total processing time for the dominant processor) for the same instance. Recall that the $Pm|line_j|C_{max}$ problem is NP-hard for the number $m \geq 5$ of processors considered in this experiment, and the optimal value is not known. Since the optimal value is no less than the lower bound, the average ratio shown in the figures *overestimates* the average gap between the C_{max} values obtained by the algorithms and the optimal one.

From the figures, we can make several observations. First, the compact algorithms (LFC and WFC) perform better than the corresponding block-based algorithms (LFB and WFB, respectively). Second, three of the algorithms (LFC, LFB, and WFC) obtain solutions that are within 5% (and in many cases, within 2-3%) of the lower bound. Furthermore, the average ratio

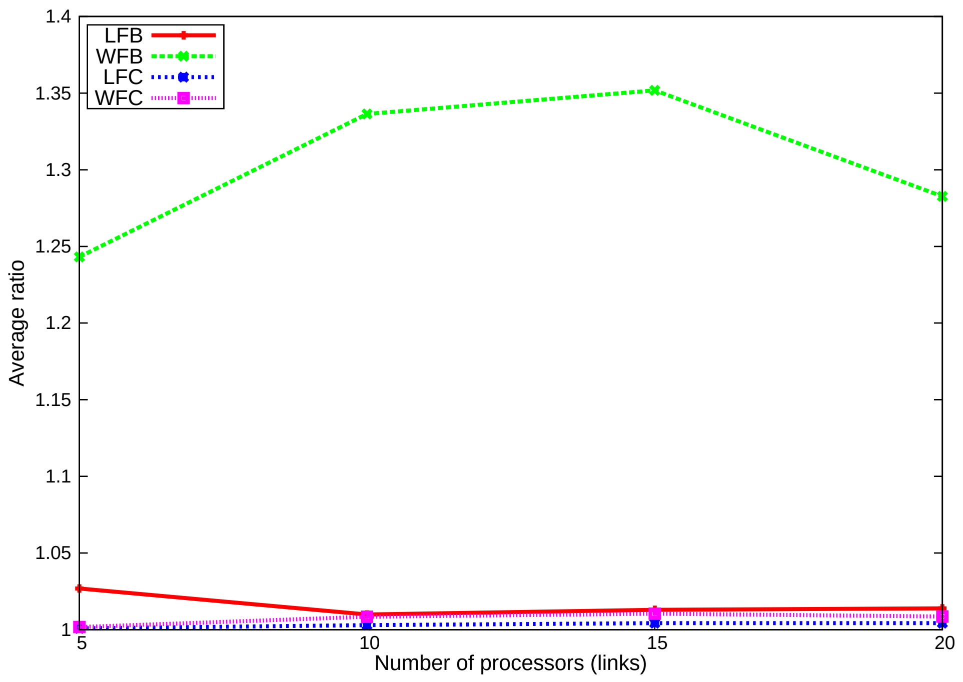


Figure 4.7: Average ratio vs. number of processors, discrete uniform distribution

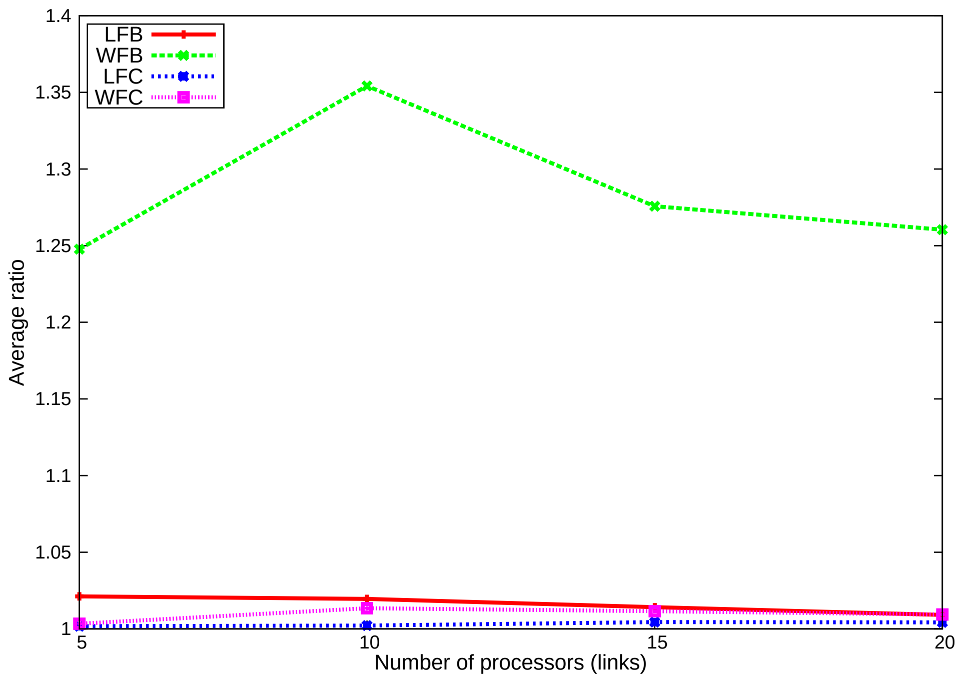


Figure 4.8: Average ratio vs. number of processors, discrete high distribution

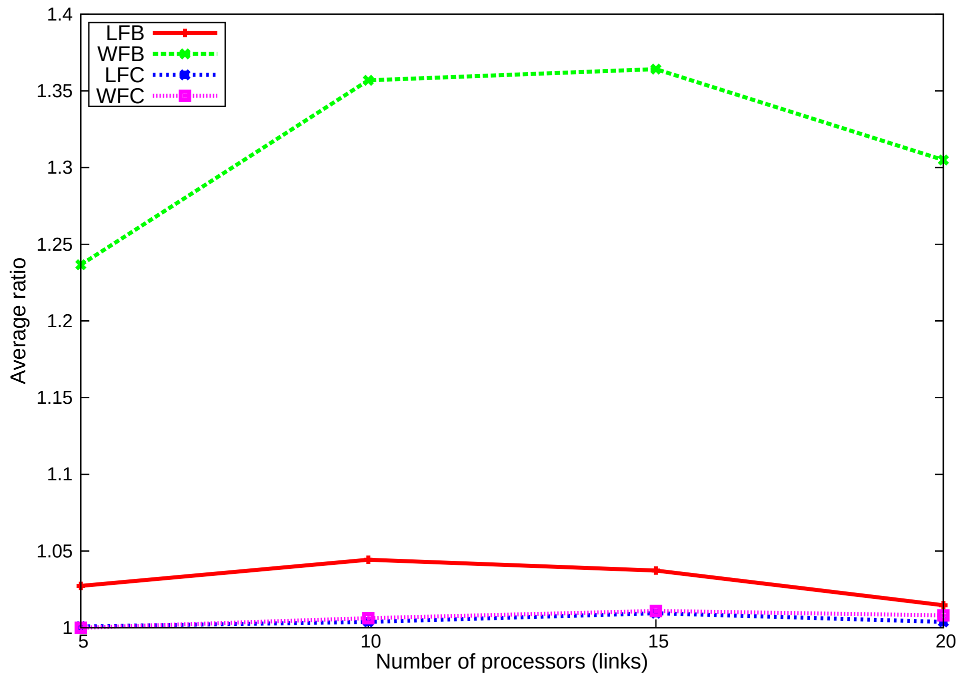


Figure 4.9: Average ratio vs. number of processors, discrete low distribution

performance of these three algorithms is not sensitive to the number of processors (chain length) or demand distribution. The block-based WFB algorithm has the worst performance over all problem instances considered in this study. This result indicates that processing tasks in the order of decreasing number of processors they require may pair short tasks with long tasks, creating large idle times within blocks. On the other hand, the WFC algorithm that processes tasks in the same order is successful in reducing these idle times, demonstrating the importance of taking into consideration the idle times in the scheduling process. Overall, these results indicate that, for problem instances representative of spectrum allocation problems arising in chains typical of the diameter of metropolitan or wide-area networks, the two compact algorithms (LFC and WFC) may obtain solutions very close to the optimal with low computational requirements.

4.5.2 Scheduling on Large Multiprocessor Systems

In this set of experiments, we consider instances of the $Pm|line_j|C_{max}$ problem applicable to large scale multiprocessor systems. Specifically, we let the number m of processors vary from 1,000 to 6,000, in increments of 1,000. For each problem instance, we generated a number of tasks equal to twice the number of processors (i.e., $n = 2m$) while the task times were selected from three distributions:

- *Uniform:* task times may take any integer value in the interval $[10, 1000]$ with equal probability.

- *Skewed High:* task times may take values in the intervals $[10, 200]$, $(200, 400]$, $(400, 600]$, $(600, 800]$ and $(800, 1000]$ with probabilities 0.10, 0.15, 0.20, 0.25, and 0.30, respectively. Once a task has been assigned to one of these intervals, it is assigned any value in that interval uniformly and randomly.

- *Skewed Low:* this is similar to the skewed high distribution, with the only difference that the probabilities that a task falls within one of the five intervals are 0.30, 0.25, 0.20, 0.15, and 0.10, respectively.

Figures 4.10-4.12 plot the average ratio achieved by the LFC, LFB, and WFC algorithms as a function of the number m of processors; each of the figures corresponds to problem instances generated by one of the three task length distributions above.

As in the earlier figures, each data point represents the average over 30 random problem instances. Similar to the results presented in the previous subsection, the WFB algorithm produces solutions with an average ratio of 1.3-1.5, substantially higher than the ones achieved by these three algorithms; hence, we omit the WFB algorithm from these figures so as to focus on the behavior of the best algorithms. We observe that all three algorithms perform very close to the lower bound: their solutions are around 1-3% away from the lower bound, on average, when $m = 1,000$, and the average ratio improves (drops) as the number of processors increases to $m = 6,000$. This behavior is consistent across all three distributions we considered for these experiments. Note that the *absolute* difference between the makespan of the solutions and the lower bound actually *increases* slowly as the number of processors increases, but the relative difference (i.e., the ratio plotted in the figures) decreases because, for a given distribution, the value of the lower bound increases with the number of processors.

Overall, our experiments indicate that the three algorithms, LFC, LFB, and WFC, perform very close to the lower bound across a range of task length distributions and number of processors, while being computationally efficient and simple to implement.

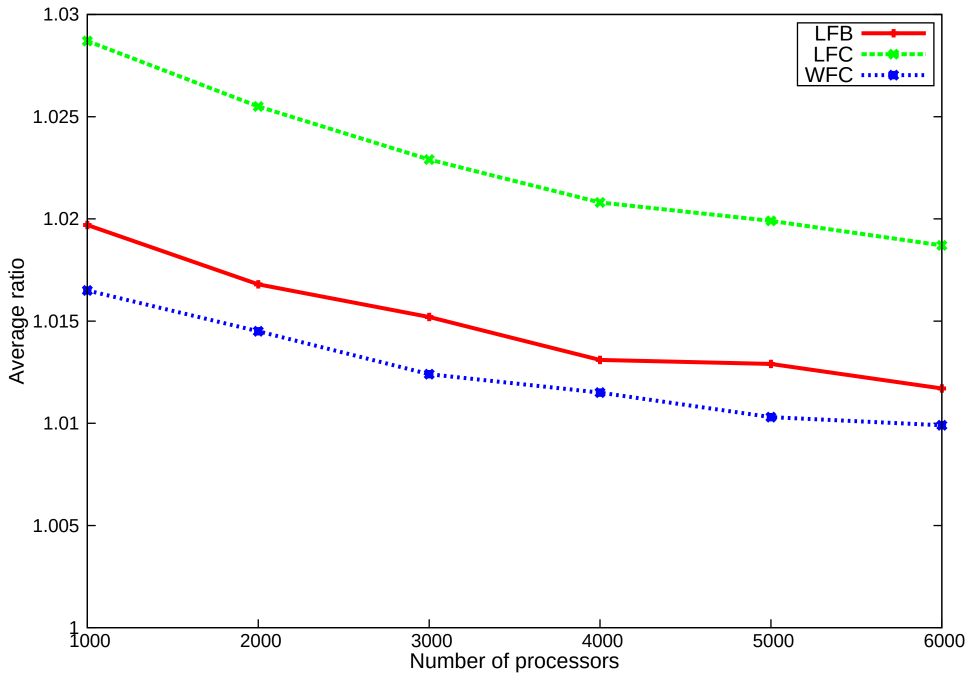


Figure 4.10: Average ratio vs. number of processors, uniform distribution

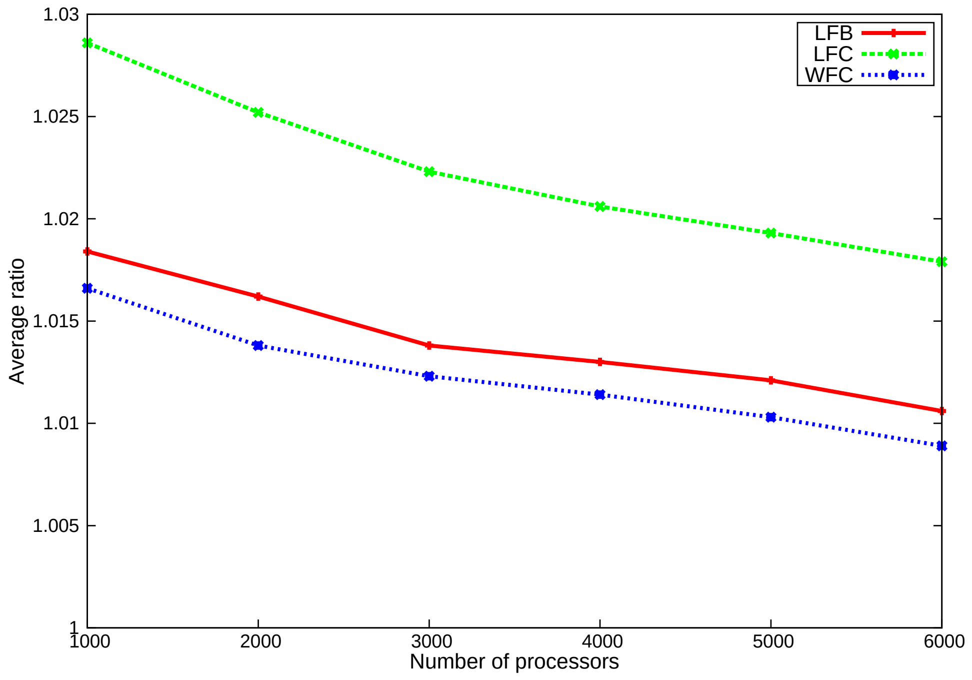


Figure 4.11: Average ratio vs. number of processors, skewed high distribution

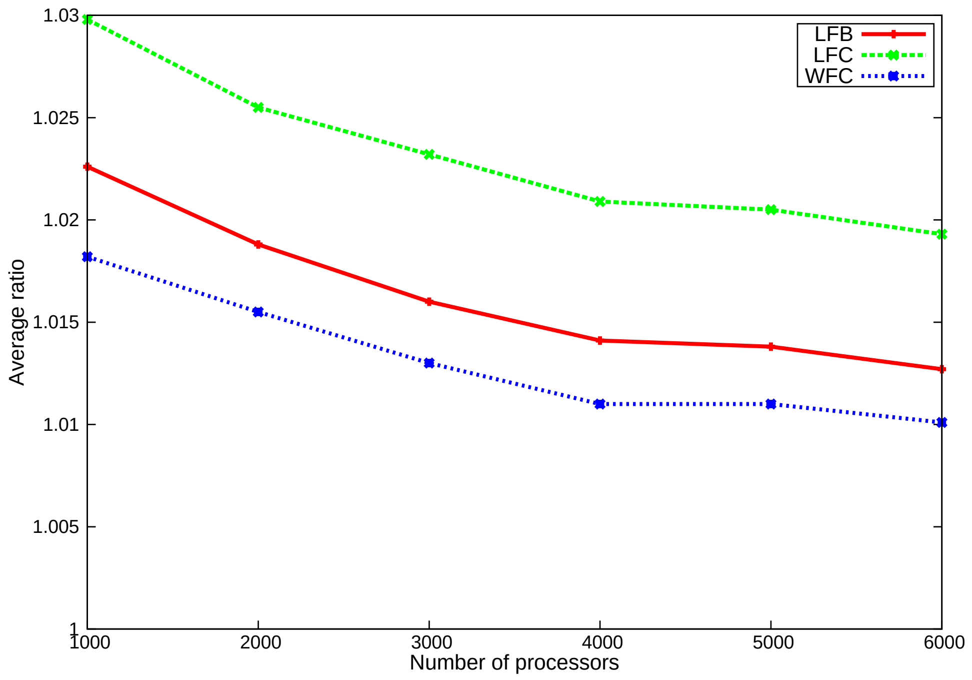


Figure 4.12: Average ratio vs. number of processors, skewed low distribution

Chapter 5

The RSA Problem in Ring EONs

In Chapter 4, we studied the SA problem in chain (path) EONs where the traffic demands are static and their routes are given in advance. We showed that this problem can be mapped to the multiprocessor scheduling problem, specifically $Pm|line_j|C_{max}$. Then, we proposed a group of heuristic and approximation algorithms for this problem.

Given the importance of SONET/SDH ring EONs, we dedicate Chapter 5 to analyze the complexity of the EONs in rings and propose approximation algorithms for this type of infrastructure under different scenarios. More specifically, we present a comprehensive study of the SA and RSA problems in bidirectional ring networks based on the derived results in Chapter 3. Building upon multiprocessor perspective, we investigate theoretically two approaches to solve the RSA problem in bidirectional rings. In Section 5.1, we consider the case of a single fixed path for each demand; this corresponds to a two-step approach for RSA in which routing of demands is performed first, followed by spectrum allocation. When each traffic demand follows a predetermined path (e.g., the shortest path) from source to destination, RSA reduces to the SA problem. We prove that the SA problem can be solved in polynomial time in small bidirectional rings, and we develop constant-ratio approximation algorithms for large rings. In Section 5.2, we show that the RSA problem, in which routing and spectrum allocation are considered jointly, is intractable for rings with as few as four nodes. Based on insight from multiprocessor scheduling theory, we also develop an approximation algorithm for ring RSA. The approximation ratios of our algorithms are strictly smaller than the best known $(4 + 2\epsilon)$ ratio presented in [20]. We also note that our results apply to the routing and wavelength assignment problem, a special case of RSA in which all demands are of equal size.

5.1 The SA Problem in Ring Networks

In this section, we study the SA problem in bidirectional rings under the assumption that each traffic demand is carried over the shortest path from its source to the destination node; we defer discussion of the RSA problem, in which the routing and spectrum assignment problems are solved jointly, to the next section. Let N be the number of nodes of the ring network. Note that, whenever N is even, there are two shortest paths between every pair of nodes that are diametrically opposite each other. In this case, we assume that one of these paths (in either the clockwise or counter-clockwise direction) is selected and is provided as input to the SA problem.

We first note that, under shortest path routing, the clockwise and counter-clockwise directions of the ring become decoupled and completely independent of each other. Consequently, the SA problem in bidirectional rings is decomposed into two disjoint subproblems, one for each direction, that can be solved separately; the subproblem in the clockwise (respectively, counter-clockwise) direction takes as input the subset of clockwise (respectively, counter-clockwise) links and the subset of demands with shortest paths along these links. It can be seen that this decomposition is optimal, in that finding the optimal solution (i.e., minimum total spectrum on any link) for each subproblem and taking the maximum of the two is an optimal solution to the original problem on the bidirectional ring. Therefore, for the remainder of this section, we will only consider the SA subproblem for the clockwise direction of the ring. Because of symmetry, the same results apply to the subproblem defined on the counter-clockwise direction, although the optimal solution may be different (e.g., because of the fact that different demands are placed on each direction).

We have shown in Chapter 3 that the SA problem in *unidirectional rings* can be transformed to a $P|fix_j|C_{max}$ problem. Moreover, in the general case, i.e., whenever there are traffic demands between any pair of nodes, the SA problem in unidirectional rings with $N = 3$ nodes transforms to the $P3|fix_j|C_{max}$ problem that is strongly NP-hard [88]. On the other hand, the SA subproblem defined on *the clockwise direction of a bidirectional ring* is a special case of the unidirectional ring problem inasmuch as its input consists of only the subset of demands that are routed in that direction. Therefore, the problem can be solved in polynomial time for small rings, and approximation algorithms with constant ratios exist, as we show next.

Since any algorithm that solves the $P|fix_j|C_{max}$ problem also solves the SA problem, in the following we will derive results for the SA problem in bidirectional rings by studying the corresponding multiprocessor scheduling problem. In our discussion, we will make use of two concepts related to $P|fix_j|C_{max}$.

Definition 5.1.1 (Compatible Tasks) *A set $\mathcal{T}$ of tasks for the $P|fix_j|C_{max}$ problem are said to be compatible if and only if their prespecified sets of processors are pairwise disjoint, i.e., $fix_i \cap fix_j = \emptyset, \forall\ i, j \in \mathcal{T}$.*

Compatible tasks may be paired with each other (i.e., they can be executed simultaneously), as they do not share any processors.

Definition 5.1.2 (Dominant Processor and Lower Bound (LB)) *Consider an instance of $P|fix_j|C_{max}$, and let $\mathcal{T}_k$ denote the set of tasks that require processor k, i.e., $\mathcal{T}_k = \{j : k \in fix_j\}$. Clearly, all the tasks in $\mathcal{T}_k$ are pairwise incompatible, hence they have to be executed sequentially. Let Π_k denote the sum of processing times of tasks that require processor k:*

$$\Pi_k \;=\; \sum_{j \in \mathcal{T}_k} p_j, \;\; k = 1, \ldots, m. \tag{5.1}$$

Then, a lower bound LB for the problem instance can be obtained as:

$$LB \;=\; \max_{k=1,\ldots,m} \{\Pi_k\}. \tag{5.2}$$

We will refer to a processor that achieves the lower bound LB as the dominant processor.

We also find two definitions of *interval graph* and *interval chromatic number* in the graph theory quite helpful to propose a constant ratio approximation algorithm for SA and RSA in rings. Hence, we extract their definitions from [97] as follows.

Definition 5.1.3 (Interval Graph) *An interval graph G is an undirected graph in which its vertices V are obtained from closed intervals of the real numbers and edges E are constructed if and only if there is an intersection between each pair of intervals (i.e. corresponding vertices).*

Definition 5.1.4 (Interval Chromatic Number) *Consider a weighted graph $G = (V, E, \omega)$, where V, E, and ω stand for the set of vertices, edges, and positive integers on V, respectively. The notation $\omega(v)$ is used here to represent the weight of a vertex v. An interval k-coloring in G is defined by a function c from V to $\{0, 1, 2, \ldots, k-1\}$ such that $c(x) - \omega(x) - 1 < k$ and if $c(x) \leq c(y)$ and $(x, y) \in E$ then $c(x) + w(x) - 1 < c(y)$. For any given vertex v, the weighted coloring function c assigns $[c(v), ..., c(v) + \omega(v) - 1]$ of $\omega(v)$ colors so that the set of assigned intervals of colors for any two adjacent vertices are disjoint. The interval chromatic number of G tries to find the smallest possible integer k, denoted by χ_{int}, such that a valid k-interval coloring of G can be constructed.*

5.1.1 Complexity Results for Rings with $N = 3, 4$ Nodes

The following two lemmas establish that, under shortest path routing, the SA problem can be solved in polynomial time in three- and four-node bidirectional rings, since the subproblems defined on the clockwise (and, hence, also the counter-clockwise) direction yield polynomial solutions. Note also that, in the special case whereby all demands are equal to one slot, $t_{sd} = 1$, the spectrum contiguity constraint becomes redundant, and the SA problem reduces to the wavelength assignment (WA) problem [19]. Consequently, these two lemmas also establish that the WA problem is solvable in polynomial time in three- and four-node rings with shortest path routing.

Lemma 5.1.1 *The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 3$ nodes and shortest path routing is solvable in polynomial time.*

Proof. In a bidirectional ring with $N = 3$ nodes, the shortest path for each demand consists of a single link. Consider the SA subproblem defined on the clockwise direction. This subproblem has three demands, each carried on exactly one of the three clockwise links of the ring. The corresponding $P3|fix_j|C_{max}$ multiprocessor scheduling problem has three tasks, each requiring exactly one of the three processors (i.e., $|fix_j| = 1, j = 1, 2, 3$). Since the tasks are pairwise compatible, they can be scheduled simultaneously. Hence, the optimal value of the total amount of spectrum required in the network (respectively, C_{max}) is equal to the maximum demand size (respectively, the maximum task processing time). ■

Lemma 5.1.2 *The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 4$ nodes and shortest path routing is solvable in polynomial time.*

Proof. In a four-node ring, the clockwise and counter-clockwise paths between two non-adjacent nodes are of equal length (i.e., two), and either may be selected as the shortest path. Let us consider the case where all demands between non-adjacent nodes are routed in the clockwise direction. In other words, for non-adjacent nodes 1 and 3, both traffic from 1 to 3 and traffic from 3 to 1 is routed clockwise; and similarly for the other pair (2,4) of non-adjacent nodes. Hence, the input to the SA subproblem consists of four one-link demands and four two-link demands. Consequently, the input to the corresponding $P4|fix_j|C_{max}$ problem consists of four single-processor tasks and four two-processor tasks. Let us denote these tasks as T_1, T_2, T_3, T_4, T_{12}, T_{23}, T_{34}, and T_{41}, where the subscript of each task denotes the processors in the corresponding set fix_j.

The proof is by construction of the optimal schedule, as shown in Figure 5.1. Specifically, first schedule the task T_{12} in parallel with the task T_{34} starting at time $t = 0$. Then, add all the single processor tasks T_1, T_2, T_3, T_4 to this initial schedule without any gaps. Finally, execute the two-processor tasks T_{23} and T_{41} as soon as both processors of each task are available. For the instance depicted in Figure 5.1, the schedule is optimal as it is equal to the lower bound determined by the sum of the processing times of tasks requiring processor 2 (the dominant

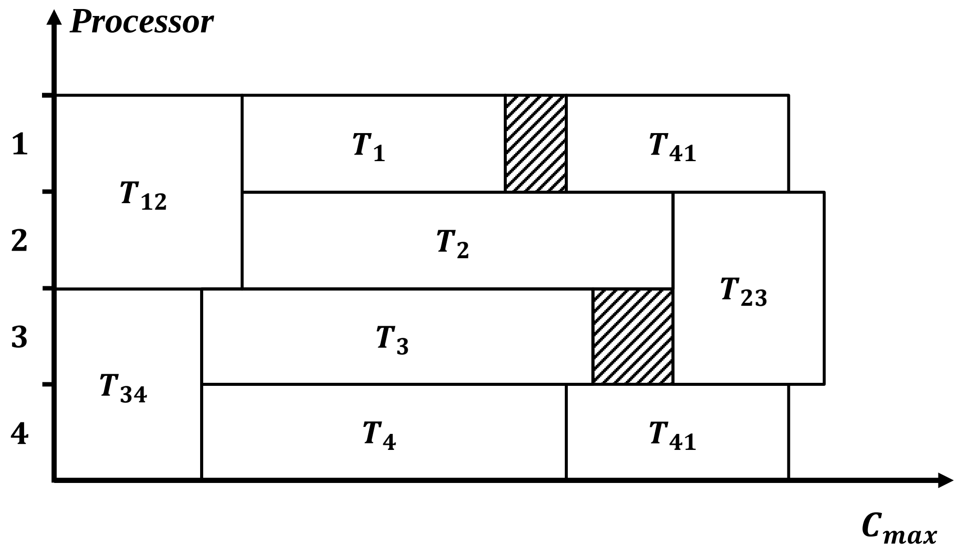


Figure 5.1: Optimal schedule for the clockwise direction of a four-node bidirectional ring with shortest path routing

processor). In fact, because of symmetry, the schedule is optimal regardless of which processor is the dominant one.

If some of the demands between non-adjacent nodes are routed in the counter-clockwise direction, then the instance of $P4|fix_j|C_{max}$ defined on the clockwise direction will not include the corresponding two-processor tasks. Again, it can be seen that the above algorithm yields an optimal schedule. For instance, if task T_{23} is excluded from Figure 5.1, then the schedule remains optimal. The same is true if T_{41} is excluded, or T_{23} and T_{41} are both excluded, or any combination of two-processor tasks is excluded. If all two-processor tasks are excluded (i.e., all demands between non-adjacent nodes are routed in the counter-clockwise direction), then the problem contains only single-processor tasks and the algorithm again produces an optimal schedule. ■

The above lemma shows that as long as traffic demands in a four-node bidirectional ring are routed along a shortest path (with ties broken arbitrarily), the SA problem is solvable in polynomial time using a simple algorithm that is linear in the number of tasks (spectrum demands). The following lemma shows that if one of the demands between adjacent nodes takes a non-shortest path, the SA problem becomes NP-complete. The proof is by reduction from the PARTITION problem definition 4.2.1.

Following standard NP-completeness proofs, the proof of the following lemma, as well as that of Theorem 5.1.1, shows that (1) there is a polynomial transformation of any instance of

PARTITION to an instance of the corresponding $Pm|fix_j|C_{max}$ problem, and (2) a partition exists if and only if an optimal schedule for the $Pm|fix_j|C_{max}$ problem also exists. Specifically, in both proofs, we include a number of gadget tasks in the instance of the $Pm|fix_j|C_{max}$ problem that are independent of the PARTITION instance, and select the C_{max} value to ensure that the second condition above is satisfied.

Lemma 5.1.3 *The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 4$ nodes and such that:*

- *all demands between non-adjacent nodes are routed in the clockwise direction, and*
- *all demands between adjacent nodes are routed along their (one-link) shortest path in the clockwise or counter-clockwise direction, except for one such demand that is directed to a three-link path in the clockwise direction,*

is NP-complete.

Proof. If a traffic demand with a one-link shortest path in the counter-clockwise direction is routed along the alternate clockwise three-link path, then the $P4|fix_j|C_{max}$ problem defined on the clockwise direction will include a three-processor task. Without loss of generality, assume that this three-processor task requires processors 3, 4, and 1 (similar arguments apply for any other three-processor task). Given an instance of PARTITION, we create an instance of this $P_4|fix_j|C_{max}$ as follows. For each $a_j \in A$ we create a task τ_j with processing time $p_j = a_j$ and $fix_j = \{2\}$ (note that these tasks must be executed by processor 2, the one that is *not* required by the three-processor task). We also create the following eight gadget tasks:

task	p_j	fix_j
T_a	B	$\{1, 2\}$
T_b	$B/2$	$\{3, 4\}$
T_c	B	$\{2, 3\}$
T_d	$B/2$	$\{4, 1\}$
T_e	$B/2$	$\{3, 4, 1\}$
T_1	B	$\{1\}$
T_2	B	$\{3\}$
T_3	$3B/2$	$\{4\}$

If A can be partitioned into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then there is a feasible schedule with $C_{max} = 3B$, as shown in Figure 5.2.

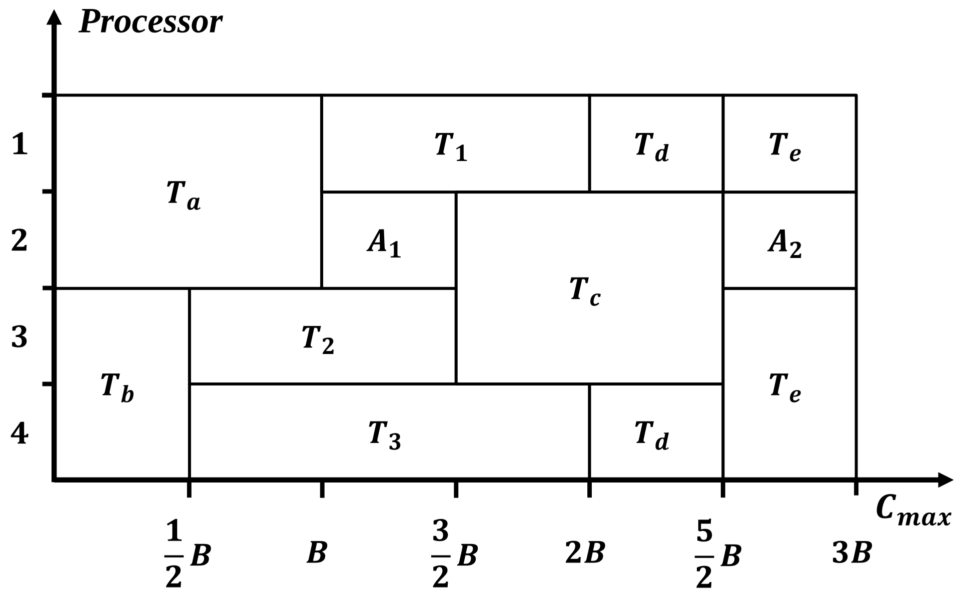


Figure 5.2: A feasible schedule with $C_{max} = 3B$ for the clockwise direction of a four-node bidirectional ring with shortest-path routing except for one demand routed along a three-link path

Conversely, let us assume that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 3B$. Without loss of generality, suppose that T_a and T_b are executed before T_c and T_d in $\mathcal{S}$; otherwise, we can use similar arguments and reach the same conclusion. Then, all the single processor tasks T_1, T_2, and T_3 must be executed immediately after T_a or T_b complete, as scheduling any other task at that time would lead to a makespan greater than $3B$. T_c must also be scheduled exactly right after T_2 and before T_e, otherwise it would not be possible to obtain the schedule with length of at most $3B$. Using a similar argument, T_d must be scheduled right after T_1 and T_3 and before T_e, and in parallel with T_c. The schedule corresponding to this set of tasks is shown in Figure 5.2 where only the intervals $[B, 3B/2]$ and $[5B/2, 3B]$ are available for the execution of the PARTITION jobs on processor 2. Therefore, a partition must exist. ■

5.1.2 Complexity Results for Rings with $N \geq 5$ Nodes

The next theorem states that the SA problem on five-node bidirectional rings (and, hence, on any larger ring) is intractable.

Theorem 5.1.1 *The SA subproblem defined in the clockwise direction of a bidirectional rings with $N = 5$ nodes and shortest path routing is NP-complete.*

Proof. As the number of nodes is odd, there is a unique shortest path for each traffic demand between any two non-adjacent nodes; therefore, the problem in the clockwise direction includes only the demands with a shortest path along the clockwise links. The proof is by reduction from the PARTITION problem, and follows an approach similar to the one we used in the proof of Lemma 5.1.3. Specifically, for each $a_j \in A$, we create a task τ_j with processing time $p_j = a_j$ and $fix_j = \{2\}$. We also create the following set of tasks:

task	p_j	fix_j
T_a	$3B/2$	$\{1, 2\}$
T_b	$5B/2$	$\{2, 3\}$
T_c	$B/2$	$\{3, 4\}$
T_d	B	$\{4, 5\}$
T_e	$2B$	$\{5, 1\}$
T_1	$3B/2$	$\{1\}$
T_2	$2B$	$\{3\}$
T_3	$7B/2$	$\{4\}$
T_4	$2B$	$\{5\}$

If there exists a partition of A into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then we can execute the tasks as shown in Figure 5.3 and create a feasible schedule with $C_{max} = 5B$.

Conversely, assume that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 5B$. Similar to the proof of Lemma 5.1.3 and without loss of generality, suppose that T_a and T_d are executed before T_c and T_e in $\mathcal{S}$; otherwise, we can use similar arguments and reach the same conclusion. We need to schedule T_2 in parallel with T_a and T_d, otherwise the schedule length will exceed $5B$. As T_d completes earlier than T_a, we need to execute T_4 before T_e. Therefore, T_1 must be scheduled right after T_a and before T_e. On the other hand, T_3 must be executed immediately after T_d, and T_c must be scheduled at the very end of $\mathcal{S}$, since if we change the order of execution of T_3 and T_c in $\mathcal{S}$, the makespan will be greater than $5B$. Finally, executing T_c between $[9B/2, 5B]$ means that T_b must be scheduled immediately after T_2. A feasible schedule corresponding to this set of tasks is shown in Figure 5.3 where only the intervals $[3B/2, 2B]$ and $[9B/2, 5B]$ are available for the execution of the PARTITION jobs on processor 2. Thus, we conclude that a partition of A must exist. ■

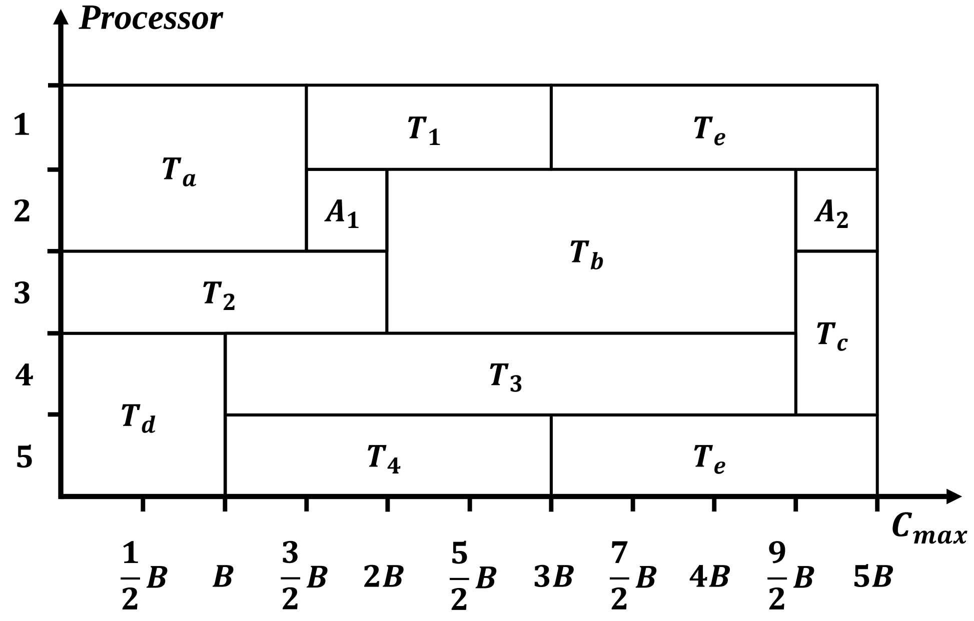


Figure 5.3: A feasible schedule with $C_{max} = 5B$ for the clockwise direction of a five-node ring with shortest path routing

5.1.3 Approximation Algorithms

In this section, we first provide approximation algorithms for the SA problem on bidirectional rings with $N = 5, 6$ and 7 nodes under shortest path routing. We then develop approximation algorithms for bidirectional rings with $N \geq 8$ nodes. Since, as we mentioned earlier, the WA problem is a special case of SA, all approximation algorithms in this section also apply to WA.

5.1.3.1 Rings with $N = 5 - 7$ Nodes

Lemma 5.1.4 *There exists an 1.5-approximation algorithm for the SA subproblem defined on the clockwise direction of a bidirectional ring with $N = 5$ nodes and shortest path routing.*

Proof. As we mentioned earlier, in a five-node ring each traffic demand has a unique shortest path. Therefore, the clockwise direction serves $10(= 5 * 4/2)$ demands, and the corresponding scheduling problem has 10 tasks as shown in Figure 5.4, where the subscript of each task indicates the processors required by the task. Without loss of generality, let processor 3 be the dominant processor, i.e., the one that achieves the lower bound LB in (5.2). Let OPT denote the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

Consider now the seven tasks that do *not* require processor 3, shown in the left part of the schedule in Figure 5.4. The scheduling problem consisting of these seven tasks can be viewed as the scheduling problem on a four-processor system (i.e., one without processor 3), similar to the one depicted in Figure 5.2 -- but with three rather than four two-processor tasks. In essence, this scheduling problem corresponds to the SA problem on the clockwise direction of the five-node after removing the link corresponding to processor 3 and the three traffic demands using that link. Based on our earlier result regarding the four-node rings, these seven tasks can be scheduled optimally, as shown on the left part of Figure 5.4. Let OPT' be the makespan of this schedule; then, $OPT' \leq OPT$.

Now consider the three tasks that require processor 3. These can be scheduled back-to-back without any gaps, as shown in the right part of Figure 5.4. The makespan of this schedule is equal to LB. Hence, the makespan of the two-part, 10-task schedule depicted in Figure 5.4 is equal to: $OPT' + LB \leq 2 \times OPT$.

We can improve the approximation ratio of 2 by modifying the above two-part schedule as follows. Without loss of generality, assume that $T_{23} \geq T_{34}$ as indicated in Figure 5.4; if T_{34} is larger than T_{23}, then simply reverse the roles in the following discussion. In this case, we have that:

$$\begin{aligned} T_{34} &\leq T_3 + T_{23} \\ \Rightarrow 2T_{34} &\leq T_3 + T_{23} + T_{34} \;=\; LB \;\leq\; OPT \\ \Rightarrow T_{34} &\leq 0.5 \times OPT \end{aligned} \tag{5.3}$$

Now slide the right part of the schedule in Figure 5.4 (i.e., the three tasks T_3, T_{23} and T_{34}) as far left as possible so that tasks T_3 and/or T_{23} overlap with the tasks in the left part of the schedule. Consider the resulting nine-task schedule, i.e., the one consisting of all tasks of the problem except T_{34}. It can be seen that this schedule is optimal for these nine tasks. Let OPT'' be the makespan of this nine-task schedule, and $OPT'' \leq OPT$. Scheduling task T_{34} immediately after the end of this schedule results in a ten-task schedule of length $OPT'' + T_{34}$. Using (5.3), we conclude that the makespan of this schedule is no larger than $1.5 \times OPT$. ■

Lemma 5.1.5 *There exist 2-approximation algorithms for the SA subproblem defined on the clockwise direction of bidirectional rings with $N = 6, 7$ nodes and shortest path routing.*

Proof. The proof is by construction of a two-part schedule similar to the one we created for the proof of Lemma 5.1.4. The proof is omitted due to its length, and the details are available in Appendix C. ■

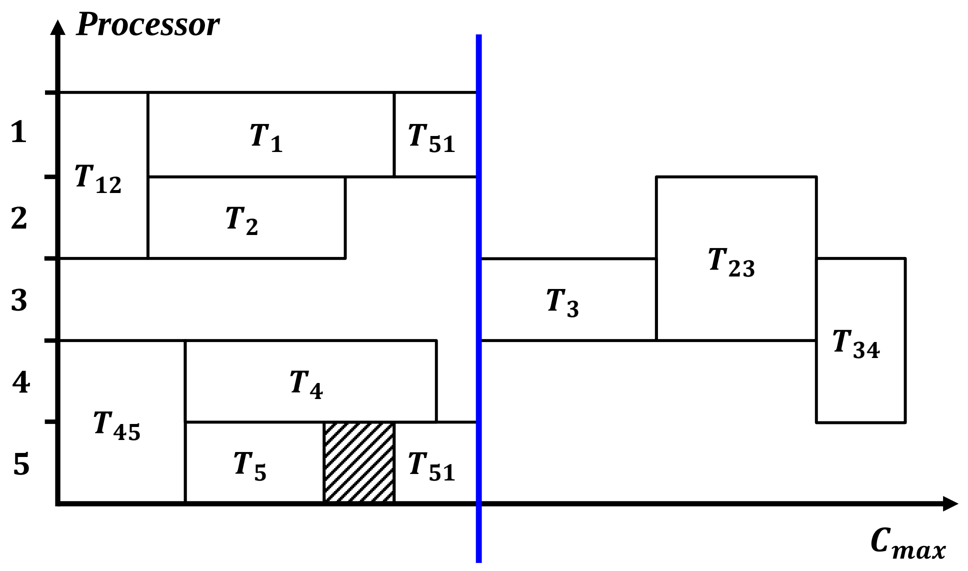


Figure 5.4: Two-part schedule for a five-node bidirectional ring with shortest path routing

5.1.3.2 Rings with $N \geq 8$ Nodes

We now present a general approximation algorithm for rings of any size. Consider the SA problem defined on the clockwise direction of a ring with $N \geq 8$ nodes and shortest path routing. The key idea is based on the observation that if we remove a link from the ring along with the traffic demands whose shortest paths use this link, the resulting SA subproblem is equivalent to the SA problem on a directed path with $N-1$ nodes. Furthermore, the $2+\epsilon$ approximation algorithm in [98] for computing the interval chromatic number of interval graphs can be used to solve the SA problem in chain networks with the same performance bound [20]. Therefore, the approximation algorithm for rings consists of the following steps:

1. Formulate the $PN|fix_j|C_{max}$ problem for the clockwise direction of the original ring.
2. Let processor N be the dominant processor (and relabel the processors appropriately if necessary).
3. Remove processor N and all tasks j that use this processor (i.e., tasks j such that $N \in fix_j$); the resulting scheduling problem corresponds to the SA problem on a $(N-1)$-link chain.
4. Use the $2+\epsilon$ approximation algorithm in [98] to create schedule $\mathcal{S}_1$ for the scheduling problem on the chain network.

5. Schedule all tasks that use processor N sequentially without any gaps to create schedule $\mathcal{S}_2$.

6. Concatenate schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ to create schedule $\mathcal{S}$ for the ring network.

Let OPT be the optimal makespan for the ring network. By construction, the makespan of $\mathcal{S}_2$ is equal to $LB \leq OPT$, while the makespan of $\mathcal{S}_1$ is no longer than $(2+\epsilon)OPT$. Hence, the approximation ratio of the above algorithm for an N-node ring is $3+\epsilon$, better than the $4+2\epsilon$ algorithm presented in [20].

5.1.4 Numerical Results

The approximation ratios of the algorithms described in the previous subsection correspond to worst-case inputs, and we expect that the algorithms will perform better on average. To investigate the average-case performance of the algorithms, we have carried out simulation experiments on rings of various sizes. We assume that the network supports the following data rates (in Gbps): 10, 40, 100, 400, and 1000. For each problem instance, we generate random traffic rates between every pair of nodes based on one of three distributions: (1) *Uniform*: traffic demands may take any of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ with equal probability; (2) *Skewed low*: traffic demands may take one of the five discrete values above with probabilities $0.30, 0.25, 0.20, 0.15$, and 0.10, respectively (i.e., the lower data rates have higher probability to be selected); or (3) *Skewed high*: traffic demands may take one of the five discrete values above with probabilities $0.10, 0.15, 0.20, 0.25$, and 0.30, respectively (i.e., the higher data rates have higher probability to be selected). Once the traffic rates between every source-destination pair have been generated, we calculate the corresponding spectrum slots as follows. We assume that the slot width is 12.5 GHz, and the 16-QAM modulation format, such that demands of size 10, 40, 100, 400, and 1000 Gbps require 1, 1, 2, 8, and 20 slots, respectively, consistent with the values used in [36, Table 1].

Since the optimal solution is not known for rings with more than four nodes, we compute the lower bound as in expression (5.2). We then compute the ratio of the makespan produced by the algorithm to the lower bound. Note that the lower bound is not tight, as it ignores any gaps introduced by the scheduling of incompatible tasks in the optimal solution. Therefore, this ratio overestimates the difference between the solution produced by the algorithm and the optimal one. Figure 5.5 plots this ratio as a function of ring size for the three demand distributions; each data point in the figure represents the average of thirty random problem instances.

As we can see, the average performance of the approximation algorithms is significantly better than what their respective constant (worst-case) ratios suggest. For instance, for a five-node ring, the algorithm is within 15% of the lower bound although its worst-case ratio is

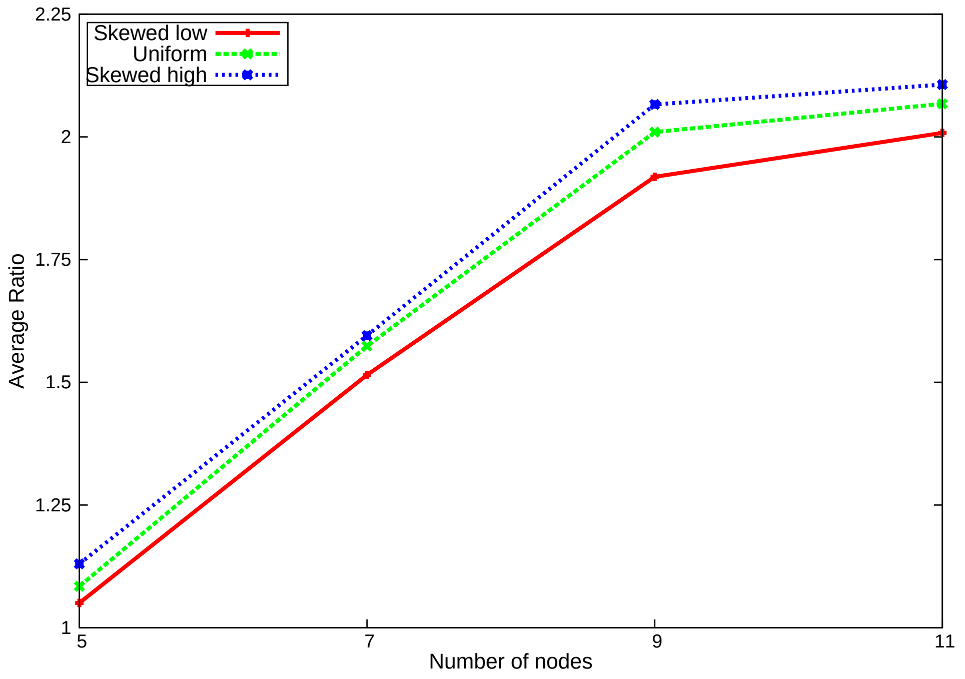


Figure 5.5: Average ratio of solutions produced by the approximation algorithms to the lower bound

1.5; whereas for a seven-node ring, the worst-case ratio is 2, but the average ratio is at most 1.6. Further, for rings of nine or more nodes, the worst-case ratio is 3, but the average ratio is around 2. Recall that the average ratio is relative to the lower bound, not the optimal, hence the actual performance of the algorithms (i.e., compared to the optimal solution) is better than the figure suggests.

Finally, we note that for the problem instances used to derive the results of Figure 5.5, the running time of the approximation algorithms was about 15 ms on a 3.10 GHz 4-core Xeon CPU; this value did not depend on the demand distributions or ring sizes used in our experiments.

5.2 The RSA Problem in Ring Networks

Let us now turn our attention to the RSA problem in bidirectional rings. Unlike the previous section where we assumed that traffic demands are routed on the shortest path, our objective is to determine both a route and a spectrum allocation for each demand. Since there are exactly two paths between each pair of nodes in a ring network, the general RSA problem on rings reduces to the RSA problem with $k = 2$ fixed-alternate paths in Definition 2.1.1.

We first present results to establish the complexity of the RSA problem in rings, followed by a new approximation algorithm. Our discussion builds upon the results of Lemma 3.2.1 which

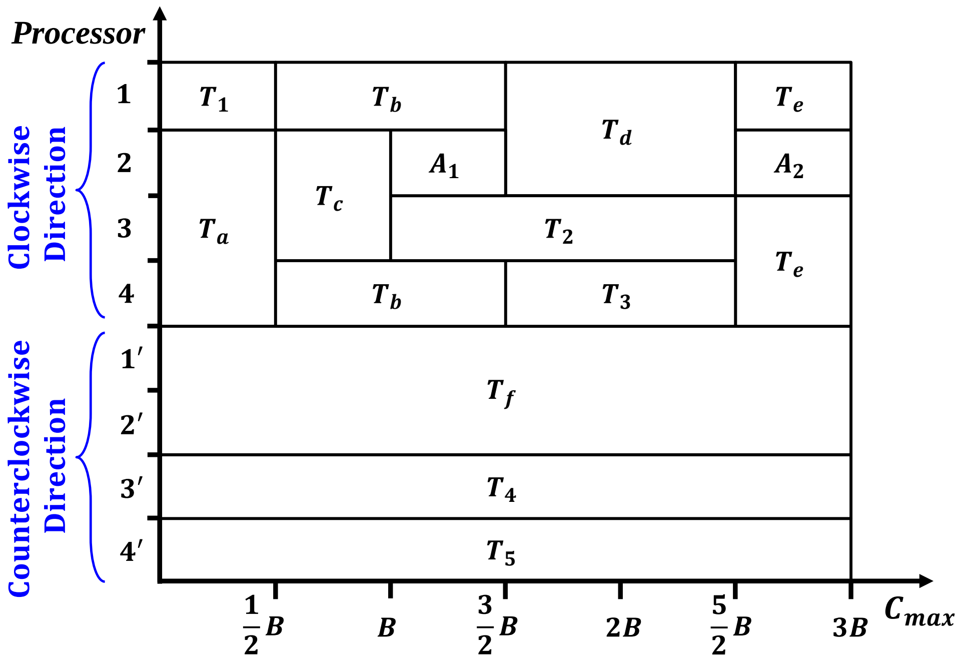


Figure 5.6: A feasible schedule for the RSA problem in a 4-node bidirectional ring with $C_{max} = 3B$

shows that the RSA problem with fixed-alternate routing is a special case of the $P|set_j|C_{max}$ multiprocessor scheduling problem.

5.2.1 Complexity Results

Lemma 5.2.1 *The RSA problem in 3-node bidirectional rings is solvable optimally in polynomial time.*

Proof. We will show that in a bidirectional ring with $N = 3$ nodes, the solution in which each traffic demand takes the shortest (i.e., one-link) path, is optimal.

Consider the corresponding $P|set_j|C_{max}$ with six processors and six tasks. Clearly, the length of the longest task is a lower bound on the optimal makespan, i.e., $OPT \geq LB = \max_{j=1,\dots,6}\{p_j\}$. In the solution to the $P|set_j|C_{max}$ problem defined by shortest path routing in the RSA instance, each task is executed on a different single processor. Hence, the tasks are pairwise compatible and may all start execution at time $t = 0$. Consequently, this solution is optimal as its makespan is equal to $\max_{j=1,\dots,6}\{p_j\}$. ■

Theorem 5.2.1 *The RSA problem in 4-node bidirectional rings is NP-complete.*

Proof. Consider a 4-node bidirectional ring with traffic demands between each pair of nodes, for a total of 12 ($= 4 \times 3$) types of demands. Let $\{1, 2, 3, 4, 1', 2', 3', 4'\}$, denote the eight directed links of the network such that l and $l', l = 1, 2, 3, 4$, are the links in the clockwise and counter-clockwise direction, respectively, between adjacent nodes in the ring. Also, assume that, in the clockwise direction, link 1 is adjacent to 2, 2 is adjacent to 3, 3 to 4, and 4 to 1, and similarly for links in the counter-clockwise direction. Each demand may be assigned a path in either the clockwise or counter-clockwise direction. For instance, a demand may take either a one-link path (say, along link 1) in the clockwise direction or the three-link path (along links $4'$, $3'$, and $2'$) in the counter-clockwise direction.

The equivalent multiprocessor scheduling problem $P8|set_j|C_{max}$ has $m = 8$ processors, which we assume are labeled identically to the corresponding links, and is constructed according to Lemma 3.2.1. We will prove that this scheduling problem is NP-complete by reduction from the PARTITION problem.

Given an instance of PARTITION, we create an instance of $P8|set_j|C_{max}$ with the eight processors $\{1, 2, 3, 4, 1', 2', 3', 4'\}$. For each $a_j \in A$, we create a task τ_j with processing time $p_j = a_j$ and $set_j = \{\{2\}, \{1', 4', 3'\}\}$; in the equivalent RSA problem, this demand may be routed either along link 2 in the clockwise direction or along the path $< 1', 4', 3' >$ in the counter-clockwise direction. We also create the following eleven gadget tasks:

task j	p_j	set_j
T_a	$B/2$	$\{\{2, 3, 4\}, \{1'\}\}$
T_b	B	$\{\{4, 1\}, \{3', 2'\}\}$
T_c	$B/2$	$\{\{2, 3\}, \{1', 4'\}\}$
T_d	B	$\{\{1, 2\}, \{4', 3'\}\}$
T_e	$B/2$	$\{\{3, 4, 1\}, \{2'\}\}$
T_f	$3B$	$\{\{3, 4\}, \{2', 1'\}\}$
T_1	$B/2$	$\{\{1\}, \{4', 3', 2'\}\}$
T_2	$3B/2$	$\{\{3\}, \{2', 1', 4'\}\}$
T_3	B	$\{\{4\}, \{3', 2', 1'\}\}$
T_4	$3B$	$\{\{4, 1, 2\}, \{3'\}\}$
T_5	$3B$	$\{\{1, 2, 3\}, \{4'\}\}$

If it is possible to partition A into A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then there exists a feasible schedule as shown in Figure 5.6 with $C_{max} = 3B$.

Conversely, suppose that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 3B$. Since T_4 and T_5 have length equal to $3B$, they must be executed on their respective single-processor set; scheduling either of them on the respective three-processor set would create conflict with some other task, resulting in a longer schedule. T_f, also of length $3B$, must be executed on its two-processor set that is compatible with the single-processor sets of T_4 and T_5. Since all four processors $1', 2', 3'$, and $4'$ are busy in the interval $[0, 3B]$ (equivalently, the counter-clockwise direction of the ring is fully utilized), the remaining tasks must be assigned to the other four processors (equivalently, the corresponding demands must be routed in the clockwise direction) to ensure that $C_{max} \leq 3B$.

Without loss of generality, assume that T_a is executed before T_e in $\mathcal{S}$; otherwise, similar arguments can be used to reach the same conclusion. T_1 is the only remaining task that is compatible with T_a and must be executed in parallel with the latter. Then, tasks T_b and T_c must be scheduled immediately after T_a and T_1 complete. Since T_c completes earlier than T_b, T_2 must be executed immediately following T_c, otherwise the schedule length for the clockwise direction would exceed $3B$. Similarly, as soon as T_b completes execution, T_d and T_3 must be scheduled in parallel. Finally, we note that the only remaining gadget task, T_e, must be appended at the end of this schedule of tasks to ensure that the makespan does not exceed $3B$. The schedule corresponding to this ordering of tasks is shown in Figure 5.6, where only the intervals $[B, 3B/2]$ and $[5/2B, 3B]$ are available for the PARTITION jobs. Thus, a partition exists. ■

5.2.2 Approximation Algorithms

The best approximation algorithm for the m-processor scheduling problem $Pm|set_j|C_{max}$ was developed in [91] and has a ratio of $m/2$. The algorithm proceeds in two phases:

1. **Processor assignment.** In the first phase, each task j is assigned to one of its alternate processor sets in set_j. Consider an assignment $\mathcal{F}$, and let $LB_{\mathcal{F}}$ denote the processing time on the dominant processor under $\mathcal{F}$, as given by expression (5.2). A dynamic programming algorithm was developed in [91] to obtain in pseudopolynomial time an optimal assignment $\mathcal{F}^\star$ such that $LB^\star = LB_{\mathcal{F}^\star}$ is minimum over all possible assignments. Clearly, $LB^\star$ is a lower bound on the optimal makespan OPT for the original $Pm|set_j|C_{max}$ problem, i.e., $LB^\star \leq OPT$.

2. **Task scheduling**. Given the optimal assignment $\mathcal{F}^\star$, the original problem reduces to a $Pm|fix_j|C_{max}$ problem in which the objective is to schedule the tasks so as to minimize the makespan. A polynomial heuristic is used to solve this problem, and it is shown that the makespan C achieved by this scheduling heuristic is such that $C \leq (m/2)LB^\star$. Hence, the two-phase algorithm is an $(m/2)$-approximation algorithm for the original $Pm|set_j|C_{max}$ problem.

The above two-phase approximation algorithm for $Pm|set_j|C_{max}$ corresponds to a natural decomposition of the RSA problem into two subproblems that are solved sequentially: a *routing* problem (in which a demand is assigned to either the clockwise or counter-clockwise path in a manner that takes into account the spectrum demands), and a *spectrum assignment (SA)* problem (in which spectrum is assigned to each demand along the path determined by the solution to the routing problem).

Note that a ring with N nodes has a total of $m = 2N$ directed links (i.e., processors in the corresponding scheduling problem). Hence, a straightforward application of the two-phase approximation algorithm to the RSA problem in rings would yield an approximation ratio of N. However, as we noted earlier, once the routing of demands has been determined, the clockwise and counter-clockwise directions of the ring become independent of each other and the corresponding SA problems may be solved separately. Therefore, rather than solving a single $P2N|fix_j|C_{max}$ problem in the second phase, it is only necessary to solve two $PN|fix_j|C_{max}$ problems, one for each direction of the ring. With this observation, the approximation ratio of the two-phase algorithm for the RSA problem in rings is $N/2$ rather than N.

We now show that it is possible to further improve the approximation ratio for the RSA problem in rings. First, we note that linear time approximation ratios for the $P4|fix_j|C_{max}$ and $P5|fix_j|C_{max}$ problems with ratios of 1.5 and 2, respectively, were developed in [99]. By using these algorithms in the task scheduling phase above, rather than the general one presented in [91], the two-phase algorithm yields approximation ratios of 1.5 and 2 for rings with $N = 4$ and $N = 5$ nodes, respectively.

For larger rings (i.e., $N \geq 6$), we leverage the approximation algorithm for the SA problem we developed in Section 5.1.3.2 to obtain a two-phase approximation algorithm for the RSA problem with a constant ratio that is smaller than $N/2$:

1. **Routing.** Use the dynamic programming algorithm in [91] to assign each traffic demand to the clockwise or counter-clockwise path.

2. **Spectrum Assignment.** Consider only the traffic demands routed along the clockwise direction and assign spectrum to them by solving the corresponding $PN|fix_j|C_{max}$ problem with the approximation algorithm in Section 5.1.3.2; repeat for the traffic demands in the counter-clockwise direction.

Following similar arguments as in Section 5.1.3.2, we conclude that the above two-phase algorithm for the RSA problem in N-node rings has an approximation ratio of $3 + \epsilon$.

Chapter 6

DA-RSA in Ring EONs

Distance adaptive spectrum allocation exploits the tradeoff between spectrum width and reach to improve resource utilization by tailoring the modulation format to the level of impairments along the path. In this Chapter, we present efficient and effective algorithms for the distance-adaptive RSA (DA-RSA) problem in rings. In Section 6.1, we introduce the general result for the DA-RSA with fixed alternate routing in general mesh networks and show this problem is a special case of a general multiprocessor scheduling problem in which a task may be executed by alternate sets of processors. Based on this transformation, we introduce a set of scheduling algorithms in Section 6.2. In Section 6.3, we present the numerical results to compare the performance of these algorithms with respect to the lower bound. Our results indicate that as the network size increases beyond a point that depends on the traffic demand distribution, the spectrum overhead associated with using a long path becomes sufficiently high that it is always best to use the shortest path.

6.1 DA-RSA in General Graphs As a Special Case of Multiprocessor Scheduling

Distance-adaptive (DA) spectrum allocation, a concept first introduced in [36], exploits the tradeoff between spectrum width and reach (for the same data rate) to improve utilization by tailoring the modulation format to the level of impairments: a high-level modulation format with narrow spectrum and low SNR tolerance may be selected for a short path, whereas a low-level modulation with a wider spectrum and high SNR tolerance may be used for a longer path [43]. The DA-RSA problem with fixed-alternate routing in mesh elastic optical networks can be defined as:

Definition 6.1.1 (DA-RSA) *Given a directed graph $G = (\mathcal{V}, \mathcal{A})$ with $\mathcal{V}$ vertices (nodes) and*

$\mathcal{A}$ arcs (directed links), k alternate routes, $r^1_{sd}, \ldots, r^k_{sd}$, from each node s to each node d, and a spectrum demand matrix $T = [t^l_{sd}]$, such that (i) t^l_{sd} is the number of spectrum slots required to carry the traffic from source s to destination d along the l-th route between the two nodes, $l = 1, \ldots, k$, and (ii) spectrum demands may increase (but not decrease) with the path length, i.e.,

$$|r^l_{sd}| \leq |r^h_{sd}| \Rightarrow t^l_{sd} \leq t^h_{sd}. \qquad (6.1)$$

select one of the k possible routes for each spectrum demand and assign spectrum slots along all the arcs of this route such that the total amount of spectrum used on any arc in the network is minimized while the following three constraints are satisfied:

- *spectrum contiguity constraint: each demand is assigned contiguous spectrum on all the arcs of each route.*
- *spectrum continuity constraint: each demand is assigned the same spectrum along all the arcs of its route.*
- *non-overlapping spectrum constraint: demands that share an arc are assigned non-overlapping parts of the available spectrum.*

If there is only one possible route for each traffic demand (i.e., $k = 1$ in the above definition), then the DA-RSA problem reduces to the distance adaptive spectrum assignment (DA-SA) problem. In Chapter 3, we have proved that the SA (and similarly DA-SA) problem in mesh elastic optical networks is a special case of the multiprocessor scheduling problem $P|fix_j|C_{max}$. That is, the SA problem can be transformed to $P|fix_j|C_{max}$, but the reverse is not always true. Based on this reduction, any algorithm that solves the $P|fix_j|C_{max}$ problem also solves the SA problem. The definition of $P|fix_j|C_{max}$ can be found in Definition 3.1.1.

Consider now the more general multiprocessor scheduling problem $P|set_j|C_{max}$, defined as follows [91, 92]:

Definition 6.1.2 (General $P|set_j|C_{max}$) *Given a set of m identical processors, a set of n tasks, a prespecified set $set_j = \{fix^1_j, \ldots, fix^k_j\}$ of k alternative processor sets to execute each task j, and processing time p^l_j for executing task j on set fix^l_j, schedule these n tasks under three constraints: (1) preemption is not allowed; (2) each task j is processed by exactly one set of processors in set_j simultaneously; and (3) each processor can execute at most one task at each time, so as to minimize the makespan $C_{max} = \max_j C_j$ of the schedule, where C_j represents the completion time of task j.*

We now show that the DA-RSA problem with fixed-alternate routing in mesh networks is a special case of general $P|set_j|C_{max}$.

Lemma 6.1.1 *DA-RSA with fixed-alternate routing in mesh networks transforms to general* $P|set_j|C_{max}$.

Proof. Consider an instance of the DA-RSA problem with fixed-alternate routing on a directed topology graph $G = (\mathcal{V}, \mathcal{A})$, a set of k routes $\{r^1_{sd}, \dots, r^k_{sd}\}$ for each source-destination pair (s, d), and demand matrix $T = [t^l_{sd}], l = 1, \cdots, k$. It is possible to build an instance of $P|set_j|C_{max}$ such that: (1) there is a processor i for every arc in $a_i \in \mathcal{A}$, (2) there is a task j for each source-destination pair (s, d), (3) there is a $set_j = \{fix^1_j, \dots, fix^k_j\}$ for each task j with $fix^l_j = \{q : a_q \in \{r^l_{sd}\}\}$ where (s, d) is the source-destination pair corresponding to task j, and (4) and processing time of task j on processor set fix^l_j is $p^l_j = t^l_{sd}, l = 1, \cdots, k$. In this transformation, each arc in the DA-RSA problem maps to a processor in the scheduling problem, each spectrum demand to a task, each alternate route of a demand to one of the alternate processor sets of the corresponding task, and the number of spectrum slots along a route of a demand to the processing time of the task on the corresponding set of processors. Note that, because of (6.1), the processing times of each task j in the $P|set_j|C_{max}$ instance will obey this relationship:

$$|fix^l_j| \le |fix^h_j| \Rightarrow p^l_j \le p^h_j. \tag{6.2}$$

The spectrum contiguity constraint in the given instance of DA-RSA is equivalent to the no preemption constraint in the constructed multiprocessor scheduling problem. The spectrum continuity constraint guarantees that all the processors within a alternate set of processors execute the corresponding task simultaneously. Finally, the non-overlapping spectrum constraint assures that a processor works at most on one task at a time. Similarly, the total amount of required spectrum on an arc of graph G in the DA-RSA problem is equivalent to the completion time of the last task executed on the the corresponding processor. Accordingly, minimizing the spectrum use on any arc of the DA-RSA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem general $P|set_j|C_{max}$. ■

We also note that the reverse of the above lemma is not true, i.e., general $P|set_j|C_{max}$ does not transform to DA-RSA and hence, it is a more general problem. The proof is by counter-example and similar to the proof of Lemma 3.2.2.

Clearly, the $P|fix_j|C_{max}$ problem is a special case of general $P|set_j|C_{max}$ where there is only one set of processors (i.e., $k = 1$) to execute each task. Therefore, once a set of processors among the $k > 1$ alternate sets is selected to execute task j, the general $P|set_j|C_{max}$ problem reduces to $P|fix_j|C_{max}$, in which case any algorithm that solves the latter problem may be applied to schedule the tasks.

6.1.1 Lower Bound for Ring Networks

In order to evaluate the performance of an algorithm for the DA-RSA problem, and since the optimal solution cannot be obtained in polynomial time, it is important to compute a lower bound (LB). To this end, we note that the amount of flow across any cut of the network is a lower bound on the amount of spectral resources that would be needed on any link. The tightest such bound occurs for a cut with the maximum flow between the two network partitions. In general, determining such a cut for a mesh network is a hard problem. In a ring network, however, we find such a cut by considering all possible two-link cuts and selecting the one with the maximum flow. In an N-node ring, there are $\frac{N!}{(N-2)!2!}$ two-link cuts, hence a lower bound can be obtained in $O(N^2)$ time. Note that an N-node bidirectional ring has N links in each direction, hence the corresponding multiprocessor scheduling problem has $m = 2N$ processors; therefore, the complexity of obtaining the lower bound can also be expressed as $O(m^2)$.

6.2 DA-RSA Algorithms for Ring Networks

In ring networks, each demand may take either the clockwise or the counter-clockwise path to the destination, hence the DA-RSA problem is equivalent to the general $P|set_j|C_{max}$ problem with $k = 2$ sets of processors for each task. It has been shown that, in the general case, there can be no constant-ratio polynomial time approximation algorithm for $P|set_j|C_{max}$ unless $P = NP$ [93]. Therefore, in order to solve the DA-RSA problem in large ring networks, new low complexity algorithms with good performance are needed.

The DA-RSA problem requires both routing and spectrum assignment decisions.One strategy is to first select one of the possible routes for each source-destination pair, and then assign the required amount of spectrum along each path. Such methods are commonly referred to as R+SA in the literature. A second approach is to make routing and spectrum assignment decisions jointly.

We now present four algorithms to solve the DA-RSA problem. The algorithms make routing and/or spectrum assignment decisions by building upon the multiprocessor scheduling perspective above. All four algorithms utilize the concept of compatible tasks to minimize the makespan, C_{max}, of the corresponding scheduling problem.

6.2.1 R+SA Algorithms

In this section, we describe two algorithms that first select the clockwise or counter-clockwise path for each demand, and then employ a multiprocessor scheduling algorithm to solve the corresponding general $Pm|fix_j|C_{max}$ problem. The algorithms only differ in how they make the routing decision, or, from the point of view of multiprocessor scheduling, how they select one of the two sets of processors on which a task is to be executed. The input to these algorithms is a list of tasks along with the two alternative set of processors and corresponding processing times.

The first algorithm simply assigns each traffic demand to its shortest path (in the scheduling problem, it assigns each task to the set with the smallest number of processors), with ties broken arbitrarily. We refer to this algorithm as SP. The second algorithm attempts to balance the spectrum demands on all the processors, and is referred to as traffic load balancing (TLB). A pseudocode description of the TLB algorithm is shown in Figure 6.1. Briefly, the algorithm processes the tasks sequentially. When processing task j, the algorithm tentatively adds the processing time of each set fix_j^l to the processing time of each processor in the set, and selects the set that results in the smallest total processing time on any processor. In essence, the algorithm ignores the simultaneous processing constraint (equivalently, the spectrum continuity constraint of DA-RSA), hence, it only considers the amount of work (load) in making a selection, not the actual schedule length.

The complexity of the TLB algorithm is determined by the running time of the two nested **for** loops within the outer **while** loop. Therefore, the running time of TLB is $O(kn)$ where n is the number of tasks in the input list and k is the maximum number of alternative processor sets for any task. Since, in the scheduling probelm corresponding to a ring network, the number of alternative sets $k = 2$, the complexity of the TLB algorithm is linear in the number n of input tasks.

Once a set of processors to execute each task has been determine by either the SP or TLB algorithms, the original general $Pm|set_j|C_{max}$ problem has been reduced to the $Pm|fix_j|C_{max}$ problem. In Chapter 4, we introduced a suite of list scheduling algorithms for solving the latter problem (i.e., for performing the spectrum assignment) in chain networks. Based on the comprehensive set of experiments reported there, the longest first compact (LFC) algorithm exhibits the best performance across various network sizes and traffic demand distributions. Therefore, we adopt the LFC algorithm to solve the $Pm|fix_j|C_{max}$ problem corresponding to ring networks. Since the running time of LFC is $O(n^2)$, it follows that the overall complexity of both the SP+LFC and TLB+LFC algorithms is also $O(n^2)$.

6.2.2 Joint Routing and Spectrum Assignment Algorithms

The two R+SA algorithms described in the previous section have low complexity and are easily implementable, as they decompose the DA-RSA problem into independent routing and spectrum assignment subproblems that are solved sequentially. The disadvantage of an R+SA approach, even in the case of the TLB algorithm that takes into account the work load on each processor (i.e., arc) is that it does not consider the possible idle times (i.e., spectrum gaps) that may occur due to the spectrum continuity constraint. Hence, the makespan of the schedule constructed by

Traffic Load Balancing Algorithm for $Pm|set_j|C_{max}$
Input: A list L of n tasks on m processors, each task j requires a prespecified set $set_j = \{fix_j^1, \ldots, fix_j^k\}$ of k alternative processor sets with its corresponding processing time $p_j = \{p_j^1, \ldots, p_j^k\}$ and $A_l = [a_1, \ldots, a_m]$ for alternative l where $a_i = 1$ if processor $i \in fix_j^l$; otherwise, $a_i = 0$
Output: A list L' of n tasks in which each task j having a processing time p_j and a set $fix_j \subseteq \{1, 2, \ldots, m\}$ of required processors

```
begin
  1.  $P \leftarrow [0, \ldots, 0]_{1\times m}$  //Initial processing load on $m$ processors
  2.  $C'_{max} \leftarrow 0$  // Expected makespan without idle times
  3.  while list $L \neq \emptyset$ do
  4.      $j \leftarrow$ first task in list $L$
  5.      Remove the task $j$ from list $L$
  6.      $fix_j \leftarrow \emptyset$  //Set of processors to execute task $j$
  7.      $p_j \leftarrow 0$  //Processing time to execute task $j$
  8.      for $z \leftarrow 1$ to $k$
  9.          $alt_z \leftarrow P + p_j^z\, A_z$
  10.         $C'_z \leftarrow$ takes the maximum value of $alt_z$
  11.         $C'_{max} \leftarrow min(C'_z)$
  12.     for $w \leftarrow 1$ to $k$
  13.         if $C'_w = C'_{max}$ then
  14.             $P \leftarrow alt_w$
  15.             $fix_j \leftarrow fix_j^w$
  16.             $p_j \leftarrow p_j^w$
  17. end while
end
```

Figure 6.1: A traffic load balancing (TLB) algorithm to select one set fix_j for executing each task j of the $Pm|set_j|C_{max}$ problem

an R+SA algorithm may be longer than necessary.

In this section, we propose two new algorithms that make routing decisions jointly with spectrum assignment. The algorithms are a variant of the well-known class of list scheduling algorithms in that they take as input a list of tasks, process the list sequentially, and build the schedule one task at a time, as they encounter the tasks in the list. However, our algorithms differ in two important points from classical list scheduling. First, since each task may be executed by alternate sets of processors, the input list contains not individual tasks, but rather task-processor set pairs, one pair for each set of processors that may execute a given task; therefore, we refer to these algorithms as *set scheduling* (SS). Second, the list is not built once at the beginning of the algorithms; rather, it is built incrementally during the execution of the algorithms, as we explain shortly.

The basic SS algorithm consists of the following logical steps:

1. **Task selection.** A subset of the input set of tasks is selected.

2. **Task ordering.** For each task selected in the first step, task-processor set pairs are created for each processor set that can execute this task. These task-processor set pairs are sorted in a list.

3. **Task scheduling.** The list is scanned and tasks are considered for inclusion in the schedule. Scheduled tasks are removed from further consideration.

4. **Iteration.** Repeat from the first step until all tasks have been scheduled.

We now describe the first three steps of the algorithm in more detail.
Task selection. This step starts with a set $\mathcal{S}$ of tasks (traffic demands) that have not been scheduled yet; initially, the set includes all n input tasks and decreases in size at every iteration as tasks are scheduled in the third step. Our goal is to identify tasks in $\mathcal{S}$ that are critical in terms of scheduling, and consider them early on. Therefore, we consider the ring network with only the traffic demands corresponding to the tasks in $\mathcal{S}$, determine the cut that results in the lower bound we discussed in the previous section, and identify the demands (tasks) that make up the maximum flow across this cut. Let $\mathcal{T} \subseteq \mathcal{S}$ denote the latter set of tasks. Since tasks in $\mathcal{T}$ contribute to the lower bound, it is important to minimize the gaps between them in the schedule. Therefore, we consider $\mathcal{T}$ as the next set of tasks to schedule.
Task ordering. For each task $j \in \mathcal{T}$ selected in the previous step, we pair it with each alternate processor set fix_j^l that can execute the task. In the case of a ring network in which the only two path options for a traffic demand are in the clockwise and counter-clockwise direction, there are only two alternate processor sets, fix_j^1 and fix_j^2, for the corresponding task. For each task, we sort its two task-processor set pairs in increasing order of the processor set size, i.e., $|fix_j^1| \leq |fix_j^2|$, with ties broken arbitrarily. Then, we sort the tasks in decreasing order of the processing time p_j^1 of their smallest processor set fix_j^1. This sorted list of task-processor set pairs, $L = [(1, fix_1^1), (1, fix_1^2), (2, fix_2^1), (2, fix_2^2), \ldots]$ is the input to the task scheduling step. With this order, tasks that have larger processing times, and hence are more critical in terms of scheduling, are considered earlier; and for a given task, the smaller processor set is considered first as it requires fewer resources (processors, or arcs) and smaller processing time (due to the distance-adaptive modulation).
Task scheduling. The input to this step is the list L of tasks from the previous step, and a partial schedule in which the last task ends at time t; initially, the schedule is empty and $t = 0$. We schedule the first task in list L to start execution at time t on processor set fix_1^1 (recall that $(1, fix_1^1)$ is the first item in list L. We then remove from the list both task-processor set pairs $(1, fix_1^1), (1, fix_1^2)$, and update the processors in set fix_1^1 as busy at time t. We scan list L to find the next task j and processor set that is compatible with fix_1^1; we schedule task j at

time t, update the processors on which it will be executed as busy, and remove all pairs with this task from the list. We continue scanning list L to find all the task-processor sets that are pairwise compatible, and schedule all these tasks to start at time t. Note that scheduling a task implies making both a routing decision (i.e., selecting one of the two processor sets of the task or route for the corresponding demand) and a spectrum assignment decision (i.e., assigning a start time to the task, or a starting spectrum slot for the corresponding demand).

Once we have reached the end of the list, we update the set $\mathcal{S}$ of unscheduled tasks that was provided as input to the task selection step by removing all the tasks that were scheduled in this step. We also update the end time of the new partial schedule to the maximum completion time of any scheduled task. We then continue to the fourth step to iterate until all tasks have been scheduled.

6.2.2.1 Pseudocode Description of the SS Algorithm

A pseudocode description of the SS algorithm is presented in Figure 6.2. The pseudocode consists of two phases. In the first phase, from Lines 5-21, we consider the critical tasks computed by the source/sink cut. Then, we select a task with a set of processors that are currently idle and schedule it to start exectution at the current time t. If some tasks in L (i.e., the set of tasks defined by the current source/sink cut) cannot be scheduled at scheduling instant t, we keep a copy of these tasks in list R. Phase two, which starts at Line 22, starts whenever no more tasks in L can be scheduled at time t. In this case, all the remaining tasks in $\mathcal{S}$ are considered to determine if some of them can be scheduled at time t. Finally, time t is updated to the next time some processors will become idle (i.e., the earliest time a scheduled task will complete execution), and this process is repeated until all the tasks in $\mathcal{S}$ are scheduled.

The running time complexity of the SS algorithm is defined by the nested **for** loop within the two nested outer **while** loops. Thus, the overall running time of the SS algorithm is $O(n^3)$, where n is the number of tasks.

6.2.2.2 The SS Algorithm with Shortest Path Routing (SS+SP)

The SS algorithm with shortest path routing (SS+SP) is a modified version of the SS algorithm that attempts to assign spectrum to as many demands as possible using the respective shortest paths (equivalently, to schedule tasks using the smallest processor sets). The input of each iteration of this algorithm is a sorted list L which is calculated using a source/sink cut, as with the basic SS algorithm. Under the SS+SP algorithm, list L is scanned to find tasks that can be scheduled at the current time t using their smallest processor sets. Tasks that cannot be scheduled on their smallesr processor sets are skipped, and scheduled later, i.e., either during the optimization phase of the algorithm (which starts in Line 22 of the pseudocode shown in

Set Scheduling Algorithm for $Pm|set_j|C_{max}$
Input: A set $\mathcal{S}$ of tasks (traffic demands) between each pair of nodes such that task j requires a prespecified set $set_j = \{fix_j^1, \dots, fix_i^k\}$ of k alternative processor sets with its corresponding processing time $p_j = \{p_j^1, \dots, p_j^k\}$
Output: A schedule of tasks, i.e., the time S_j when each task j starts execution on the multi-processor system

```
begin
  1. $t \leftarrow 0$   //Scheduling instant
  2. $F \leftarrow \{1, \dots, m\}$  //Set of currently idle processors
  3. $\mathcal{T} \leftarrow$ a subset of $\mathcal{S}$ determined by the source/sink cut algorithm
  4. $L \leftarrow$ a sorted list of $\mathcal{T}$
  5. while $\mathcal{S} \neq \emptyset$
  6.     $R \leftarrow \emptyset$   // List of tasks remaining from each iteration
  7.     while list $L \neq \emptyset$ do
  8.         $j \leftarrow$ first task in list $L$;    Remove task $j$ from list $L$
  9.         $fix_j \leftarrow \emptyset$
  10.        for $z = 1$ to $k$
  11.            if $fix_j^z \subseteq F$ then
  12.                $fix_j \leftarrow fix_j^z$;   $p_j \leftarrow p_j^z$;    $F \leftarrow F \setminus fix_j$
  13.                $S_j \leftarrow t$   // Task $j$ starts execution at time $t$
  14.                Remove task $j$ from $\mathcal{S}$
  15.                break
  16.        if $fix_j = \emptyset$ then Add task $j$ to list $R$
  17.    end while
  18.    $\mathcal{T} \leftarrow$ a subset of $\mathcal{S}$ determined by the source/sink cut algorithm
  19.    $L \leftarrow$ a sorted list of $\mathcal{T}$
  20.    if $L = R$ then   // Performs optimization for all the remaining tasks
  21.        $L \leftarrow$ a sorted list of $\mathcal{S}$
  22.        while list $L \neq \emptyset$ or $F \neq \emptyset$ do
  23.            $l \leftarrow$ first task in list $L$;   Remove task $l$ from list $L$
  24.            for $z = 1$ to $k$
  25.                if $fix_l^z \subseteq F$ then
  26.                    $fix_l \leftarrow fix_l^z$;    $p_l \leftarrow p_l^z$;    $F \leftarrow F \setminus fix_l$
  27.                    $S_l \leftarrow t$   // Task $l$ starts execution at time $t$
  28.                    Remove task $j$ from $\mathcal{S}$
  29.                    break
  30.        end while
  31.        $i \leftarrow$ the first task executing at time $t$ to complete
  32.        $t \leftarrow S_i + p_i$
  33.        $F \leftarrow F \cup fix_i$
  34.end while
  35.return the task start times $S_j$
end
```

Figure 6.2: A set scheduling algorithm for $Pm|set_j|C_{max}$

Figure 6.2) or after the time t is updated.

6.3 Numerical Results

We now describe the experiments we have carried out to compare the performance of the four DA-RSA algorithms in bidirectional ring networks with $N = 4, 6, 8, 10, 12, 14, 16$ nodes (recall also that the scheduling problem corresponding to an N-node bidirectional ring has $m = 2N$ processors). We generate traffic demands between each pair of nodes in the ring based on one of the following three distributions:

- *Distance-independent:* traffic demands may take any of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ with equal probability; these values correspond to data rates (in Gbps) to be supported by EONs.
- *Distance-increasing:* traffic demands may take one of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ such that higher values are assigned to a node pair with a probability that *increases* with the length of the shortest path between the two node.
- *Distance-decreasing:* traffic demands may take one of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ such that higher values are assigned to a node pair with a probability that *decreases* with the length of the shortest path between the two node.

In our experiments, we also used various other probability values for both the discrete low and discrete high distributions, but the trends regarding the relative performance of the algorithms were very similar to the ones shown below.

We consider distance adaptive spectrum allocation based on the traffic rate and the length of each possible path (i.e., number of processors in the corresponding scheduling problem) [7,36]. Thus, we assume that the slot width is 12.5 GHz, and there are two modulation formats as represented in [36]:

- 16-QAM modulation format for paths with up to 8 links (i.e., processors) such that 10, 40, 100, 400, and 1000 Gbps take 1, 1, 2, 8, and 20 slots, respectively.
- QPSK modulation format for more than 8 links (i.e., processors), whereby 10, 40, 100, 400, and 1000 Gbps are assigned 1, 2, 4, 16, and 40 spectrum slots, respectively.

The performance metric we consider in this study is the ratio of the spectrum required by the solution constructed by one of the algorithms, over the lower bound (computed as described earlier); the closer this ratio is to 1.0, the better the performance of the algorithm in terms of its use of available spectrum. As we discussed in Chapter 5, the $P2N|set_j|C_{max}$

problem corresponding to a bidirectional ring network with N nodes is NP-hard for $N \geq 4$, so the optimal makespan value is not known for the problem instances considered in this study. Clearly, this optimal value is greater than or equal to the estimated lower bound; therefore, the performance of the algorithms with respect to the optimal may be better than this ratio indicates. Nevertheless, this metric accurately characterizes the relative performance of the algorithms.

Figures 6.3-6.5 plot the average ratio of the four algorithms, denoted by SP+LFC, TLB+LFC, SS, and SS+SP, as a function of the number of ring nodes; each figure presents results for problem instances generated using the distance-independent, distance-increasing, and distance-decreasing demand distributions, respectively. Each data point on these plots is the average of ten replications, each replication being the average over 30 randomly generated instances; 95% confidence intervals, estimated using the method of batch means, are also shown in the figures.

We first observe that the best algorithm has a ratio of no more than 1.15, i.e., it is always within 15% of the lower bound on the amount of spectrum required to route all demands. Since the (unknown) optimal solution will generally lie above the lower bound, these results indicate that our algorithms are effective in constructing solutions close to optimal one.

Another important observation is that of the two R+SA algorithms, TLB+LFC outperforms SP+LFC, regardless of the demand distribution, for small- and medium-size ring networks, but SP+LFC performs better for rings with 14 or more nodes. Note that in the ring networks in which TLB+LFC is better than SP+LFC, and based on the modulation formats we consider, a demand requires the same number of slots regardless of whether it is routed on the shortest or non-shortest path. In this case, the TLB is successful in making efficient use of the spectrum resources by occasionally using non-shortest paths to balance the traffic load. However, whenever demands routed along the non-shortest path require a larger number of slots than along the shortest path, it is more difficult to achieve load balancing by using the longer path. Therefore, SP+LFC is the better solution in large networks since selecting the non-shortest path incurs a spectrum penalty along a large number of links.

We also observe that from small- to medium-size rings, the SS algorithm is able to find solutions using non-shortest paths that outperform both R+SA algorithms, but does not work well for larger ring networks. On the other other hand, the SS+SP algorithm, which gives preference to shortest paths, has by far the best performance as the ring size increases. Overall, our results indicate that (1) due to the spectrum penalty of long paths, strategies that give preference to shortest path routing work best for large rings, and (2) the SS-based algorithms outperform the R+SA algorithms across the range of ring network sizes and traffic demand distributions that we have considered in these experiments.

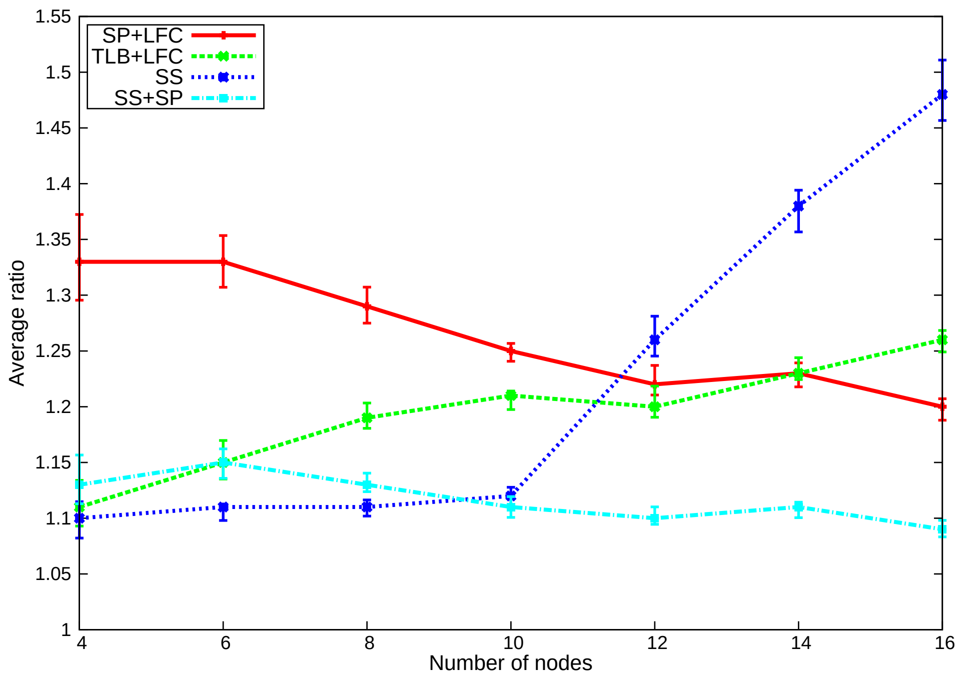


Figure 6.3: Average ratio vs. number of nodes, distance-independent distribution

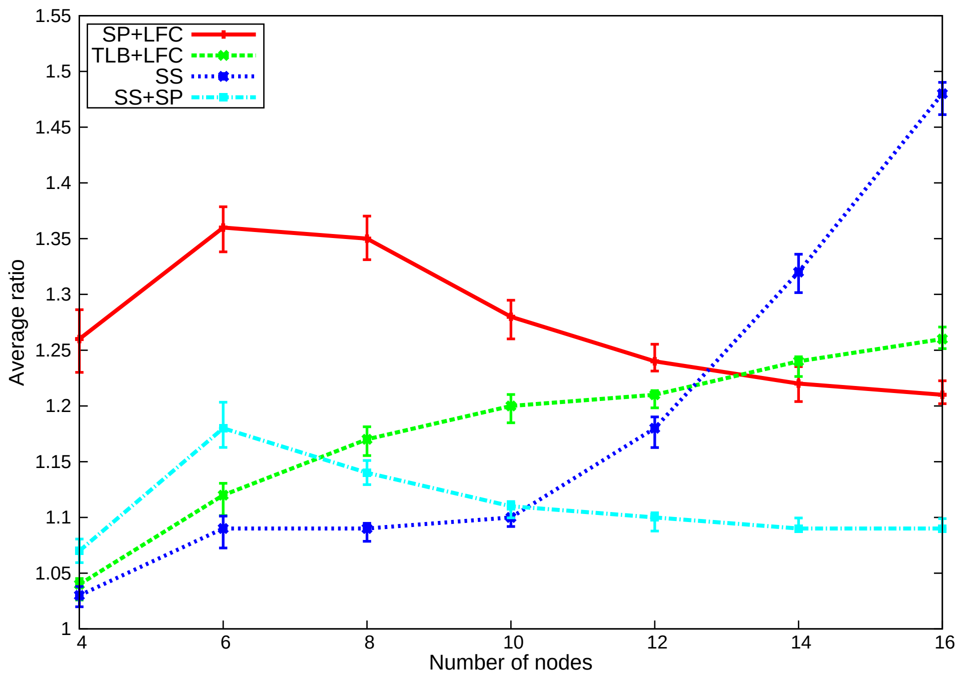


Figure 6.4: Average ratio vs. number of nodes, distance-increasing distribution

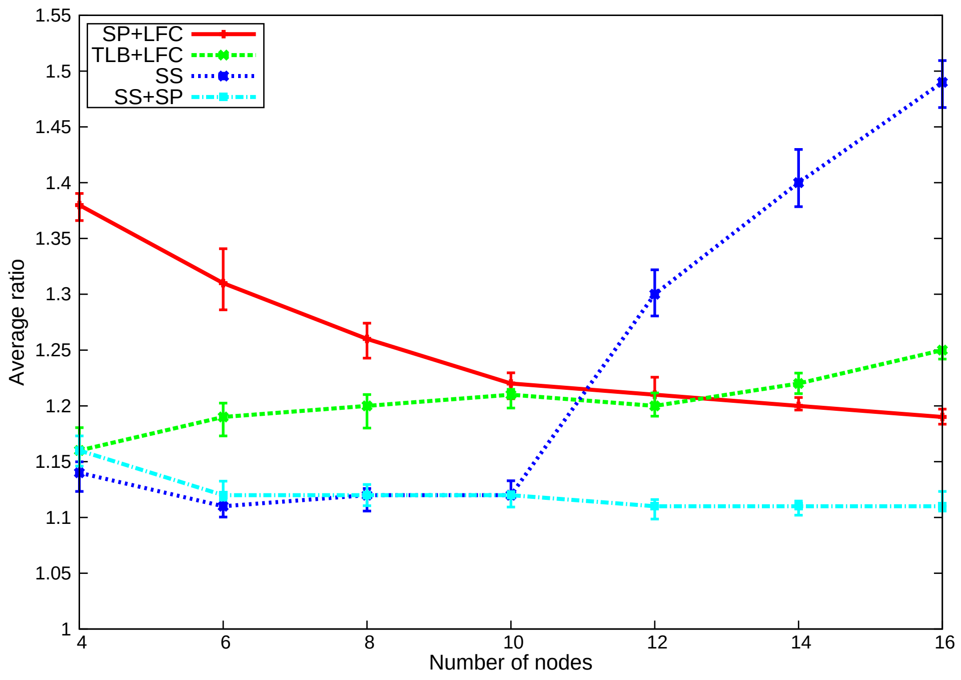


Figure 6.5: Average ratio vs. number of nodes, distance-decreasing distribution

Chapter 7

DA-RSA in Mesh EONs

As backbone and regional networks evolve from ring to mesh, optimal solutions for the RSA problem in general topologies are becoming important to network designers and operators. Hence, we study the DA-RSA problem in mesh EONs in this chapter. In Section 7.1, we show that DA-RSA in mesh networks is a special case of scheduling multiprocessor tasks on dedicated processors. Next, we discuss the complexity of the DA-RSA in mesh networks and derive some theoretical results in Section 7.2. Then, we develop a computationally efficient algorithm that builds upon list scheduling concepts to jointly tackle the routing and spectrum assignment aspects of DA-RSA in Section 7.3 and represent numerical results in Section 7.4.

7.1 DA-RSA in Mesh Networks

In Chapter 6, we have shown that the DA-RSA problem with fixed alternate routing in mesh networks transforms to the $P|set_j|C_{max}$ multiprocessor scheduling problem, but the reverse is not true. In other words, DA-RSA is a special case of $P|set_j|C_{max}$, and hence, any algorithm for the latter problem may also solve the former.

In the transformation, each arc in the DA-RSA problem maps to a processor in the scheduling problem, each traffic demand to a task, and the number of spectrum slots (respectively, the set of alternate routes) of a demand to the processing time (respectively, the set set_j of alternate processor sets) of the corresponding task, and the maximum number of spectrum slots used on any link to the makespan of the schedule. Accordingly, minimizing the maximum spectrum allocation on any arc of the DA-RSA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|set_j|C_{max}$. Furthermore, the spectrum contiguity constraint of DA-RSA is equivalent to the no-preemption constraint of $P|set_j|C_{max}$, the spectrum continuity constraint maps to the constraint that all required processors must execute a task simultaneously, and the non-overlapping spectrum constraint maps to the constraint that a processor work on at most one task at a time.

7.2 Complexity Results for Mesh Networks

The problem $P2|set_j|C_{max}$ in which the number of processors is fixed to $m = 2$, is NP-hard [94]. Moreover, it has been shown that, unless $P = NP$, no constant-ratio polynomial time approximation algorithm exists for the general problem $P|set_j|C_{max}$ [93]. However, since the DA-RSA problem is a special case of $P|set_j|C_{max}$, it is possible that polynomial or approximation algorithms exist for special topologies or spectrum demand matrices. In this section, we present theoretical results on the complexity of the DA-RSA problem.

Before we proceed, we introduce two definitions. First, we let K_N denote a *complete digraph* with N nodes. Since every pair of distinct nodes in K_N is connected by a pair of distinct arcs, one in each direction, the total number of arcs in the graph is equal to $N(N-1)$. Second, in the context of multiprocessor scheduling, we refer to tasks as *compatible* if they can be executed simultaneously, i.e., if the processor sets assigned to the tasks are pairwise disjoint. We now have the following lemma.

Lemma 7.2.1 *The DA-RSA problem on complete digraphs $K_N, N \geq 2$, is solvable in polynomial time.*

Proof. From Lemma 6.1.1 the multiprocessor scheduling problem instance corresponding to a DA-RSA instance on K_N contains $N(N-1)$ processors, one for each arc of K_N. Let us select the shortest path (i.e., direct arc) for each spectrum demand in the DA-RSA instance. Then, each task in the scheduling instance is to be executed on its own distinct processor. Therefore, the problem reduces to that of scheduling a set of single-processor tasks that are pairwise compatible. Since all tasks may be executed in parallel, the makespan of the schedule is equal to the processing time of the longest task. Recall that all instances of $P|set_j|C_{max}$ constructed from an instance of the DA-RSA problem are such that processing times of tasks satisfy expression (6.2). Therefore, this makespan is optimal. ■

Although DA-RSA may be solved optimally on a complete digraph using shortest path routing, as the above lemma implies, the next three results show that DA-RSA on general topologies derived by deleting arcs from a complete digraph, is NP-complete.

Theorem 7.2.1 *DA-RSA on a digraph G obtained by deleting the two arcs*[1] *between any pair of nodes of K_4, is NP-complete.*

[1]In typical telecommunication networks, two nodes are directly connected using two links, one in each direction. Hence, in this and the next theorem, we only consider the case of removing both arcs between a pair of nodes. It is possible to extend the results to the case of deleting one arc at a time.

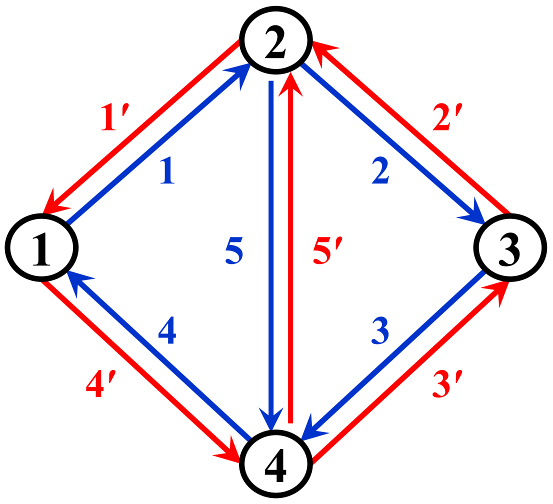


Figure 7.1: The digraph K_4' obtained after removing the two arcs between nodes 1 and 3 from K_4

Proof. Consider the four-node digraph K_4' obtained from K_4 after removing the two arcs between nodes 1 and 3, as shown in Figure 7.1; note that, because of symmetry, the proof holds if the arcs between any pair of nodes of K_4 are removed. Let $\{1, 2, 3, 4, 5, 1', 2', 3', 4', 5'\}$ represent the ten arcs of this network, as labeled in the figure. Following the transformation described in Lemma 6.1.1, we transform an instance of DA-RSA on digraph K_4' to an instance of $P|set_j|C_{max}$ with $m = 10$ processors, and we represent each processor using the same label as the corresponding arc of K_4'.

Let $\mathcal{P}_4$ represent the set of $P|set_j|C_{max}$ instances corresponding to DA-RSA instances defined on digraph K_4'. By construction, each DA-RSA instance on K_4' transforms to a unique instance of $\mathcal{P}_4$, and therefore, the reverse is also true, i.e., each instance of $\mathcal{P}_4$ transforms back to a unique instance of DA-RSA on K_4'. We now show that the scheduling problem $\mathcal{P}_4$ is NP-complete; since $\mathcal{P}_4$ transforms to DA-RSA on K_4', the latter problem is NP-complete as well. The proof is by reduction from the PARTITION problem which be can be found in Definition 4.2.1.

Given an instance of PARTITION, we create an instance of $\mathcal{P}_4$ with the ten processors labeled $\{1, 2, 3, 4, 5, 1', 2', 3', 4', 5'\}$, as shown in Figure 7.2. Specifically, for each $a_j \in A$, we create a task τ_j with processing time $p_j = a_j$ and $set_j = \{\{2\}, \{5, 3'\}, \{1', 4', 3'\}\}$. Furthermore, we create the eleven tasks listed in Table 7.1:

Each task created for the $\mathcal{P}_4$ instance corresponds to a demand between a pair of nodes in the DA-RSA problem on G. For instance, consider task T_a. Referring to Figure 7.1, task T_a corresponds to a demand from node 1 to node 4 in G, and the three alternate sets of processors for the task (i.e., the sets in the third column of the first row in Table 7.1) correspond to the three paths from 1 to 4 in G, i.e., $< 4' >$, $< 1, 5 >$, and $< 1, 2, 3 >$, respectively. Also, since

Table 7.1: Tasks created in the transformation of PARTITION $\mathcal{P}_4$

task j	p_j	set_j
T_a	B	$\{\{4'\},\{1,5\},\{1,2,3\}\}$
T_b	$B/2$	$\{\{1'\},\{5,4\},\{2,3,4\}\}$
T_c	B	$\{\{3'\},\{5',2\},\{4,1,2\}\}$
T_d	$B/2$	$\{\{2'\},\{3,5'\},\{3,4,1\}\}$
T_e	$3B$	$\{\{3,4\},\{2',1'\},\{2',5,4\},\{3,5',1'\}\}$
T_f	$3B$	$\{\{1,2\},\{4',3'\},\{1,5,3'\},\{4',5',2\}\}$
T_1	$B/2$	$\{\{1\},\{4',5'\},\{4',3',2'\}\}$
T_2	$3B/2$	$\{\{3\},\{2',5\},\{2',1',4'\}\}$
T_3	B	$\{\{4\},\{5',1'\},\{3',2',1'\}\}$
T_4	$5B/2$	$\{\{5\},\{2,3\},\{1',4'\}\}$
T_5	$3B$	$\{\{5'\},\{4,1\},\{3',2'\}\}$

the processing time of each task is independent of the set of processors on which the task is executed, condition (6.2) is satisfied.

If set A can be partitioned into A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then there exists a feasible schedule for the $\mathcal{P}_4$ problem as shown in Figure 7.2 with $C_{max} = 3B$. This schedule is also optimal since its makespan is equal to the processing time of the longest task.

Conversely, suppose that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 3B$. Since tasks T_5, T_e, and T_f have length equal to $3B$, they must be executed in parallel, i.e., they must be assigned to processor sets that are pairwise disjoint. Specifically, assigning T_5 on either of its two-processor sets would create a schedule of length longer than $3B$, hence, it must be executed on processor $5'$, as shown in Figure 7.2. As a result, T_e and T_f *must* be scheduled on processor sets $\{2',1'\}$ and $\{4',3'\}$, respectively. Given this assignment, the five processors $1', 2', 3', 4'$, and $5'$ are busy in the interval $[0, 3B]$, Therefore, the remaining tasks must be executed on processors 1, 2, 3, 4, and 5 to guarantee that the length of the schedule $C_{max} \leq 3B$.

We also note that the processing time of T_4 is $5B/2$, so this task *must* be executed on processor 5 to ensure that the makespan does not exceed $3B$. Consequently, T_b is the only task that can be scheduled on the processor set $\{4, 5\}$ to keep $C_{max} \leq 3B$. In turn, this observation implies that task T_a *must* be executed on the set $\{1, 2, 3\}$.

Without loss of generality, suppose T_a is executed before T_d in schedule $\mathcal{S}$; otherwise, similar arguments can be used to reach the same conclusion. Among the remaining tasks compatible with T_a, T_b is the only one that *must* be executed in parallel with T_a to yields a makespan of no more than $3B$. Next, tasks T_3 and T_4 *must* be scheduled immediately after the completion of

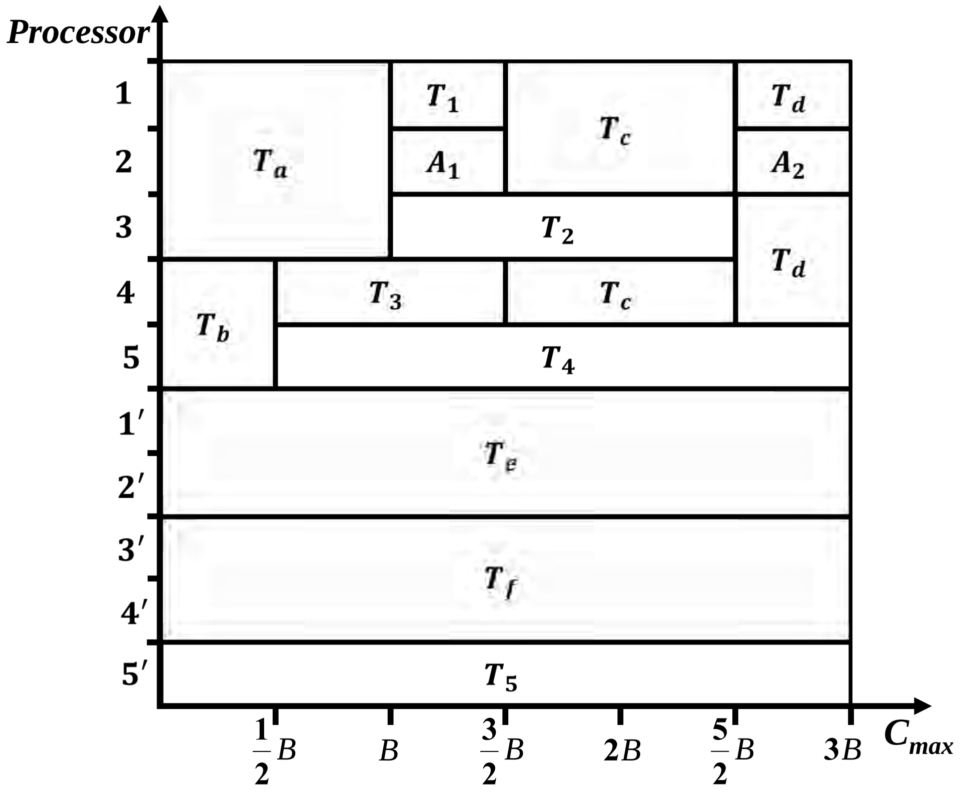


Figure 7.2: A feasible schedule with $C_{max} = 3B$ for the $\mathcal{P}_4$ problem instance of Theorem 7.2.1

T_b. As T_a completes at time B, earlier than T_3, tasks T_1 and T_2 *must* be scheduled right after T_a, otherwise the schedule length would be greater than $3B$. Similarly, as soon as T_1 and T_3 complete at time $3B/2$, T_c *must* be executed in parallel with T_2 and T_4. Finally, we observe that T_d *must* start at time $5B/2$ to ensure that the makespan does not exceed $3B$. The corresponding schedule of tasks is shown in Figure 7.2, and is such that only the intervals $[B, 3B/2]$ and $[5/2B, 3B]$ can be used to execute the PARTITION jobs. Thus, a partition of set A, i.e., a solution to the PARTITION problem, exists. ■

Theorem 7.2.1 shows that removing the two arcs between any pair of nodes of K_4 renders the DA-RSA problem on the resulting graph K_4' NP-complete. The following theorem shows that removing two pairs of arcs from K_5 yields a problem that is also NP-complete.

Theorem 7.2.2 *DA-RSA on a digraph K_5' obtained by deleting the two arcs between any two pairs of nodes of K_5, is NP-complete.*

Proof. The two pairs of nodes in K_5 whose arcs are removed may or may not have one node in common. We investigate each of these cases separately.

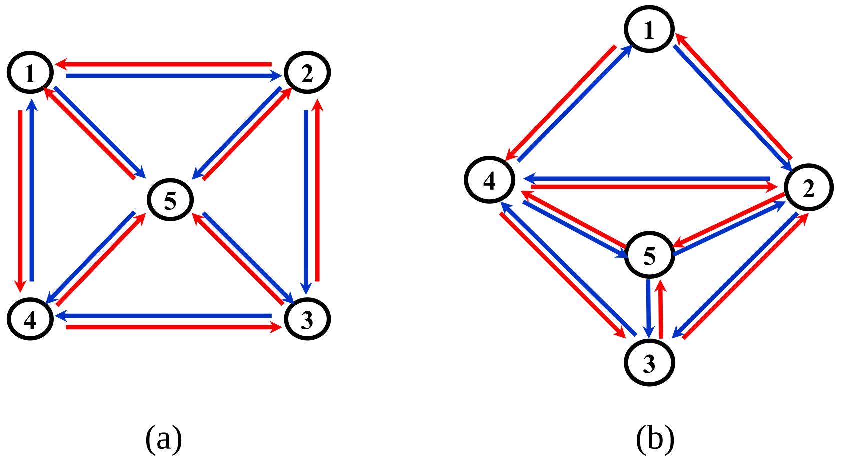


Figure 7.3: The digraph K_5' created from K_5 by removing two pairs of arcs (a) with no common node, and (b) with one common node.

Figure 7.3(a) illustrates the digraph K_5' after removing the two arcs between nodes 1 and 3, and nodes 2 and 4. Consider a DA-RSA instance in which spectrum demands to and from node 5 are very large, i.e., $t_{5j}^l = t_{j5}^l = M, j = 1, \ldots, 4$, where M is a large number. In this case, an optimal solution must be such that (1) all these large demands must be routed across the corresponding direct arc, and (2) the arcs that carry the large demands are not used to carry any other traffic. The DA-RSA problem for the remaining node pairs $(i, j), i, j = 1, \ldots, 4, i \neq j$, is equivalent to the DA-RSA on a four-node bidirectional ring, which, as we mentioned above, is NP-complete.

Similarly, if we remove the two arcs between nodes 1 and 5 and nodes 1 and 3, the result will be the digraph K_5' shown in Figure 7.3(b). Let the spectrum demands $t_{5j}^l = t_{j5}^l = M, j = 2, \ldots, 4$, be very large. As we observed in the previous case, these large demands must use the direct arcs, and the latter may not be used to carry other traffic. DA-RSA for the remaining demands is defined on a digraph identical to the one in Figure 7.1, a problem we proved to be NP-complete in Theorem 7.2.1. ■

We now provide the following complexity result for the DA-RSA problem on general graphs.

Lemma 7.2.2 *Let G be a digraph. If either K_4' or K_5' is a vertex-induced subgraph of G, then the DA-RSA problem on G is NP-complete.*

Proof. Let K_4' be a vertex-induced subgraph of G; identical arguments apply when K_5' is a vertex-induced subgraph of G. Consider an instance of DA-RSA on G with the following spectrum demands: (1) arbitrary, between nodes of the K_4' subgraph, (2) equal to a large number

M, between adjacent nodes not in the K_4' subgraph, and (3) equal to zero, between non-adjacent nodes not in the K_4' subgraph. Similar to the observations in the previous theorem, in the optimal solution, each arc of G that is not part of the subgraph K_4' only carries the traffic between the directly connected nodes. Hence, this instance reduces to a DA-RSA sub-problem on digraph K_4', which, according to Theorem 7.2.1, is NP-complete. ■

7.3 A List Scheduling Algorithm for $Pm|set_j|C_{max}$

In this section, we propose a list scheduling (LS) algorithm for the $Pm|set_j|C_{max}$ problem. Since DA-RSA is a special case of $Pm|set_j|C_{max}$, this algorithm can be used to solve the DA-RSA problem in networks of general topology. This is accomplished in three steps: (1) the DA-RSA instance at hand is first be transformed to an instance of $Pm|set_j|C_{max}$ following the process described in Lemma 6.1.1, (2) the LS algorithm is applied to construct a schedule that solves the scheduling instance, and (3) the schedule is transformed back to a solution of the DA-RSA instance.

The input to the LS algorithm is a list of tasks L, along with their corresponding k alternate sets of processors. Tasks in the list are sorted in decreasing order of the processing time on their smallest processor set; ties are broken by the size (i.e., the number of processors) of their smallest processor set, and further ties are broken arbitrarily. For each task, its alternate processor sets are sorted in increasing order of their size.

At each scheduling instant t, the algorithm scans the list L to find the first task j and processor set S_j^l that is compatible with the tasks already executing at this time t. This set S_j^l of processors is selected to execute task j starting at time t, and the algorithm removes the task from L. The algorithm updates the set of free processors at time t, and continues scanning list L, repeating the above process until no other compatible task is found. Then, the algorithm advances t to the earliest time $t' > t$ at which one of the currently executing tasks will be completed, releases the set of processors assigned to the just completed task, and repeats the above actions for time t'. The algorithm continues in this manner until all tasks in list L have been scheduled.

A pseudocode description of the LS algorithm is provided in Figure 7.4. Both the outer and inner **while** loops of the algorithm take at most $O(n)$ time, in the worst case, where n is the number of tasks in the scheduling problem. Both **for** loops take time $O(k)$ in the worst case, where k is the number of alternate processor sets. Therefore, the running time complexity of the LS algorithm is $O(kn^2)$. Since the number of tasks corresponds to the number of spectrum demands, the complexity of the algorithm when applied to the DA-RSA problem is $O(kN^4)$, where N is the number of nodes and k the number of alternate paths.

List Scheduling Algorithm for $Pm|set_j|C_{max}$
Input: A list L of n tasks on m processors, each task j defined by the set $set_j = \{fix_j^1, \ldots, fix_j^k\}$ of k alternative processor sets on which it may be executed, and the corresponding processing times $\{p_j^1, \ldots, p_j^k\}$
Output: A schedule of tasks, i.e., the time T_j when task j starts execution, along with the set fix_j of processors assigned to it and the corresponding processing time p_j

```
begin
  1.  $t \leftarrow 0$   //Scheduling instant
  2.  $F \leftarrow \{1, \ldots, m\}$  //Set of currently idle processors
  3.  while list $L \neq \emptyset$ do
  4.      $j \leftarrow$ first task in list $L$
  5.      $fix_j \leftarrow \emptyset$   //Set of processors to execute task $j$
  6.      $p_j \leftarrow 0$   //Processing time of task $j$
  7.      for $z \leftarrow 1$ to $k$
  8.          if $fix_j^z \subseteq F$ then
  9.              $fix_j \leftarrow fix_j^z$
  10.             $p_j \leftarrow p_j^z$
  11.             $T_j \leftarrow t$
  12.             $F \leftarrow F \setminus fix_j$
  13.             Remove the task $j$ from list $L$
  14.             break
  15.     while not at the end of list $L$ or $F \neq \emptyset$ do
  16.         $i \leftarrow$ first task in list
  17.         for $w \leftarrow 1$ to $k$
  18.             if $fix_i^w \subseteq F$ then
  19.                 $fix_i \leftarrow fix_i^w$
  20.                 $p_i \leftarrow p_i^w$
  21.                 $T_i \leftarrow t$
  22.                 $F \leftarrow F \setminus fix_i$
  23.                 Remove the task $i$ from list $L$
  24.                 break
  25.     end while  // no more tasks may start at time $t$
  26.     $j \leftarrow$ the first task executing at time $t$ to complete
  27.     $t \leftarrow T_j + p_j$
  28.     $F \leftarrow F \cup fix_j$
  19. end while
end
```

Figure 7.4: A list scheduling (LS) algorithm to select one set fix_j and its corresponding processing time p_j to execute each task j of the $Pm|set_j|C_{max}$ problem.

7.4 Numerical Results

We have evaluated the performance of the LS algorithm by carrying out simulation experiments with a large number of DA-RSA problem instances. Each problem instance is characterized by three parameters: (1) the network topology, (2) the number k of shortest paths for each source-destination pair, and (3) a randomly generated spectrum demand matrix.

7.4.1 Topology and Shortest Paths

In our evaluation study, we have used three general topology networks of varying size and average nodal degree:

- the 14-node, 42-arc (directed link) NSFNet shown in Figure 7.5,
- the 32-node, 108-arc GEANT2 topology depicted in Figure 7.6, and
- the 60-node, 154-arc network topology illustrated in Figure 7.7 and adapted from CORONET CONUS [100].

We used Yen's algorithm [101] to compute the the k loop-less shortest paths, $k = 1, \ldots, 7$, between each pair of nodes in each topology. Yen's algorithm takes time $O(N^3)$, where N is the number of nodes. For the experiments we present in this section, we assumed that all links have unit weight for purposes of computing shortest paths.

7.4.2 Spectrum Demand Matrix

For each DA-RSA problem instance we randomly generate a spectrum demand matrix in two steps: traffic demand generation and distance-adaptive spectrum allocation.

7.4.2.1 Traffic Demand Generation

We assume that the elastic optical network supports the following data rates (in Gbps): 10, 40, 100, 400, and 1000. Therefore, in the first step, traffic rates between every pair of nodes are drawn from one of three probability distributions:

- *Distance-independent:* each value in the set $\{10, 40, 100, 400, 1000\}$ is selected with equal probability.
- *Distance-increasing:* the probability assigned to each value in the set $\{10, 40, 100, 400, 1000\}$ depends on the length of the shortest path between the source and destination nodes, such that the probability of higher values in the set *increases* with the length of the shortest path.

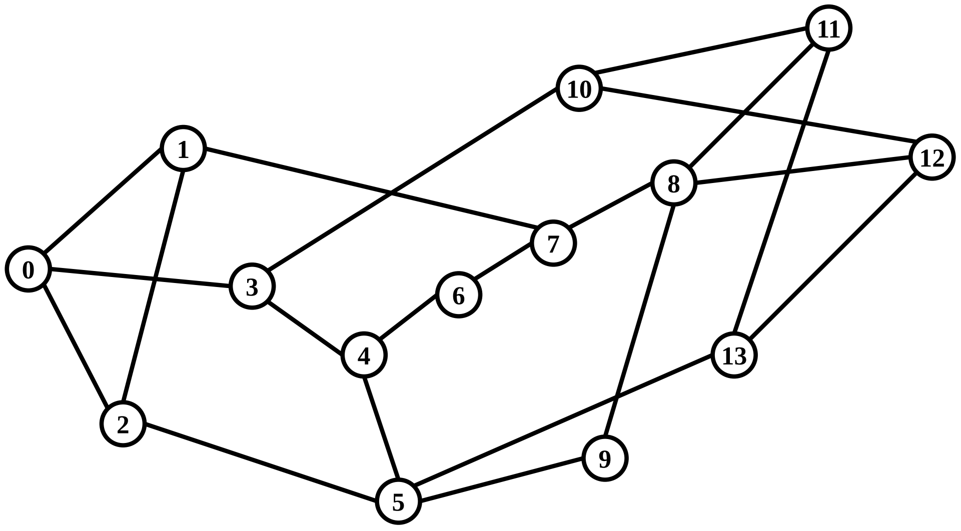


Figure 7.5: The NSFNet topology

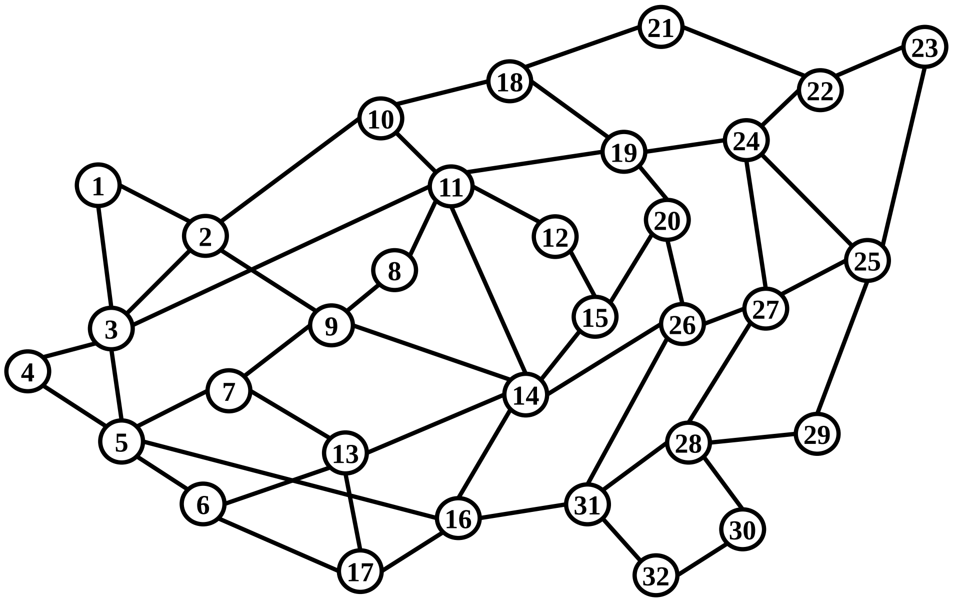


Figure 7.6: The GEANT2 topology

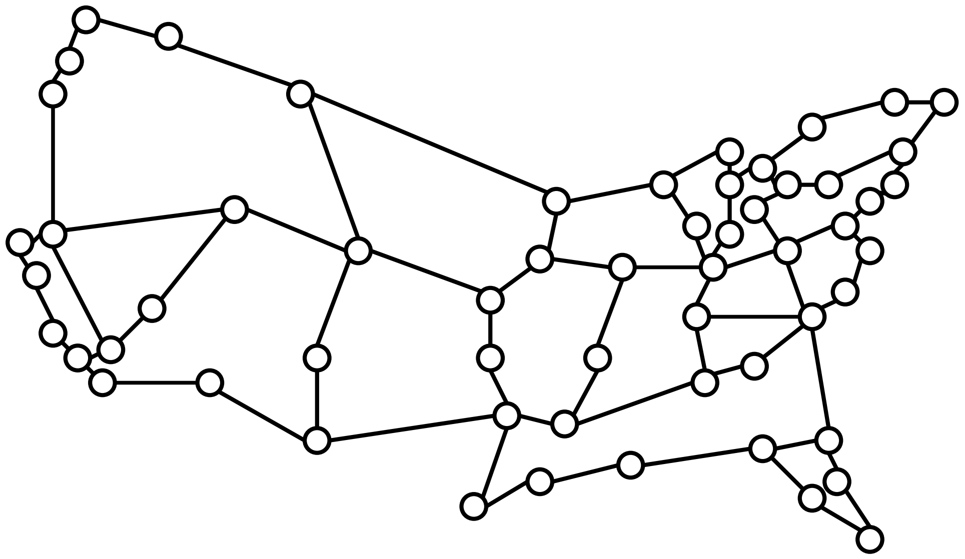


Figure 7.7: The 60-node network topology derived from the CORONET CONUS

- *Distance-decreasing:* the probability assigned to higher values in the set $\{10, 40, 100, 400, 1000\}$ *decreases* with the length of the shortest path between the source and destination nodes.

7.4.2.2 Distance-Adaptive Spectrum Allocation

In the second step, we determine the number t^l_{sd} of spectrum slots required for the traffic demand to be carried on the l-th alternate path, $l = 1, \ldots, k$, from source s to destination d. In distance-adaptive spectrum allocation, the number of slots depends on both the data rate and the length of the path [7,36]. We adopt the parameters of the study in [36], and assume a slot width of 12.5 GHz and three modulation formats:

- *Paths with up to 4 links:* the 64-QAM modulation format is used such that data rates of 10, 40, 100, 400, and 1000 Gbps require 1, 1, 2, 6, and 14 spectrum slots, respectively.
- *Paths with 5-9 links:* the 16-QAM modulation format applies, such that rates of 10, 40, 100, 400, and 1000 Gbps are assigned 1, 1, 2, 8, and 20 slots, respectively.
- *Paths with 10 or more links:* the QPSK modulation format is utilized, and data rates of 10, 40, 100, 400, and 1000 Gbps are allocated 1, 2, 4, 16, and 40 spectrum slots, respectively.

7.4.3 Evaluation Metrics

The first metric we consider is the maximum number of spectrum slots on any link in the network required by the solution to a DA-RSA problem instance obtained by the LS algorithm.

We denote this value as $MaxSlots_{LS}$; as the reader may recall, this value is equivalent to the length of the schedule constructed by the LS algorithm for the corresponding scheduling problem instance. This metric can provide insight into the impact of the number k of alternate paths or the traffic rate distribution on the use of spectrum resources in the network.

In order to evaluate the quality of the LS algorithm, and since the optimal solution cannot be obtained in polynomial time, it is important to compute a lower bound (LB). Let D_q^{in} and D_q^{out} denote the in- and out-degrees of node q. A simple lower bound for the DA-RSA problem can be calculated as follows:

$$LB \;=\; \max\{\max_s \sum_d t_{sd}/D_s^{out}, \max_d \sum_s t_{sd}/D_d^{in}\} \tag{7.1}$$

where t_{sd} in the above expression is the spectrum demand for the traffic from s to d along the shortest path between the two nodes. The metric we use to characterize of the LS algorithm is the ratio

$$R = MaxSlots_{LS}/LB. \tag{7.2}$$

Clearly, $R \geq 1.0$; the closer R is to 1.0, the better the performance of the algorithm. We note, however, that the lower bound in (7.1) only considers spectrum demands in and out of each node, and does not account for the interaction of these demands along the links of the network; therefore, we expect the bound to be loose.

The figures we present in the next section report average values for either $MaxSlots_{LS}$ or R. Specifically, each data point on these figures is the average of 10 replications of a random experiment; in turn, each replication is the average of 30 random instances generated for the stated parameters (i.e., topology, number k of paths, and traffic rate distribution). The figures also report 95% confidence intervals which can be seen to be narrow.

7.4.4 Results and Discussion

The three Figures 7.8-7.10 plot the maximum number of spectrum slots, $MaxSlots_{LS}$, as a function of the number k of alternate paths, for the NSFNet, GEANT2, and 60-node topologies, respectively. Each figure includes three curves, each representing results for problem instances with spectrum demand matrices generated by the distance-independent, distance-increasing, and distance-decreasing distributions, respectively.

We first observe that the amount of spectrum increases with the size of the network, reflecting the corresponding increase in traffic demands due to the larger number of source-destination pairs. Nevertheless, the overall behavior of the curves is consistent across the three traffic distributions and network topologies. Specifically, the amount of spectrum resources is high for shortest path routing ($k = 1$), but drops sharply (between 20-50%, depending on the distribution and topology) when demands may be routed along one of $k = 2$ alternate paths. As the number k of alternate paths increases further, the number of spectrum slots decreases more slowly and eventually levels off, indicating the diminishing returns of employing each additional path.

A final observation from the three Figures is that the solution to the DA-RSA problem is highly sensitive to the traffic demand distribution. Specifically, everything else being equal, the distance-increasing distribution requires more spectrum than the distance-independent distribution, which in turn is more resource-intensive than the distance-decreasing distribution. This result can be explained by the fact that demands between nodes that are far away from each other consume more spectral resources in the network than the same demands between two nearby nodes due to (1) the larger number of links in the paths they travel, and (2) the wider spectrum that is required to carry the demand if the length of its path crosses the threshold into a lower-level modulation with high SNR tolerance.

Let us now turn our attention to the three Figures 7.11-7.13 which plot the average ratio R in expression (7.2) against the number k of paths for each of the three network topologies; again, each figure includes three plots, one per demand distribution. Note that the lower bound in (7.1) is independent of the number k of alternate paths for each demand. Since the number of required slots, $MaxSlots_{LS}$ decreases with k, as seen in the previous three figures, we expect R to decrease as well, and this is exactly what we observe in Figures 7.11-7.13.

Nevertheless, there is an important difference between the three figures that plot the absolute value of spectrum slots required and the ones that show the average ratio. Specifically, we observe that there are significant gaps between the various curves in each of Figures 7.8-7.10, which, as we explained above, are due to the combined effects of the demand distribution and distance-adaptive spectrum allocation. On the other hand, the curves of the various distributions in Figures 7.11-7.13 are closer to each other and the average ratios of the three distributions converge to similar values. Recall that the lower bound in (7.1) depends on the demands in and out of each node in the network, and hence it depends on the traffic distribution. Therefore, the behavior of the curves in Figures 7.11-7.13 is a strong indication that, for the topologies and distributions we considered in this study, the LS algorithm is capable of exploiting alternate paths to construct solutions that move towards the lower bound, regardless of the absolute value of spectrum slots required in each problem instance.

Finally, we note that the average ratio of the LS algorithm increases with the size of the network, from around 1.8 for the NSFNet to around 2.7 for GEANT2 and about 5 for the 60-node network (these ratio values are for the largest number of alternate paths shown in the figures). This increase is partly due to the heuristic nature of the LS algorithm: as the size of the problem increases, the size of the solution space increases exponentially whereas the set of solutions examined by the algorithm increases polynomially, hence the probability of finding good quality solutions decreases. However, we argue that a significant part of the increase in

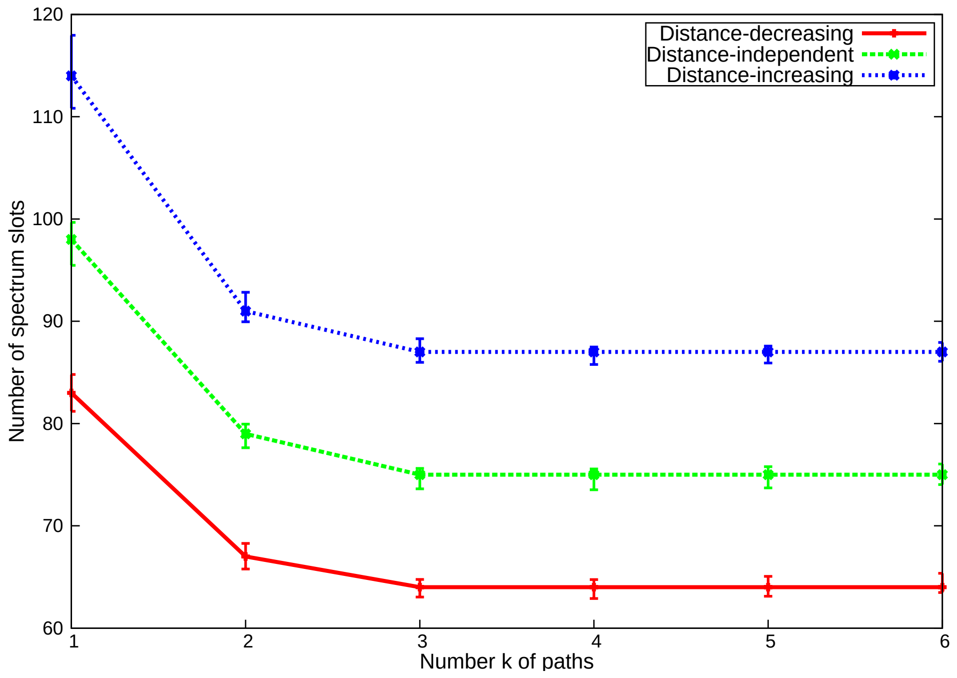


Figure 7.8: Spectrum slots vs. number k of paths in NSF

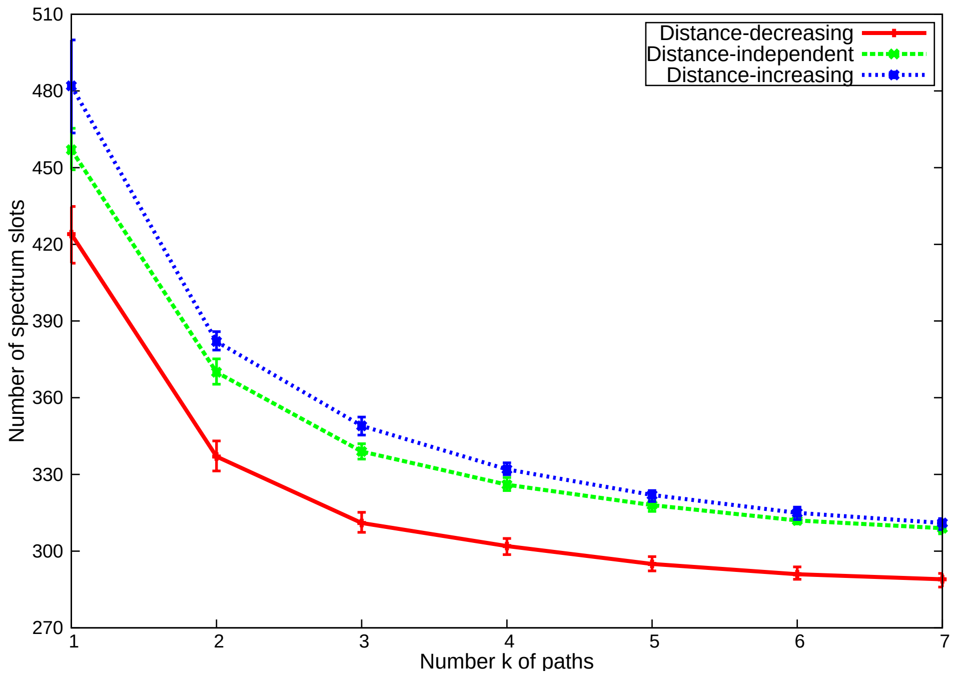


Figure 7.9: Spectrum slots vs. number k of paths in GEANT2

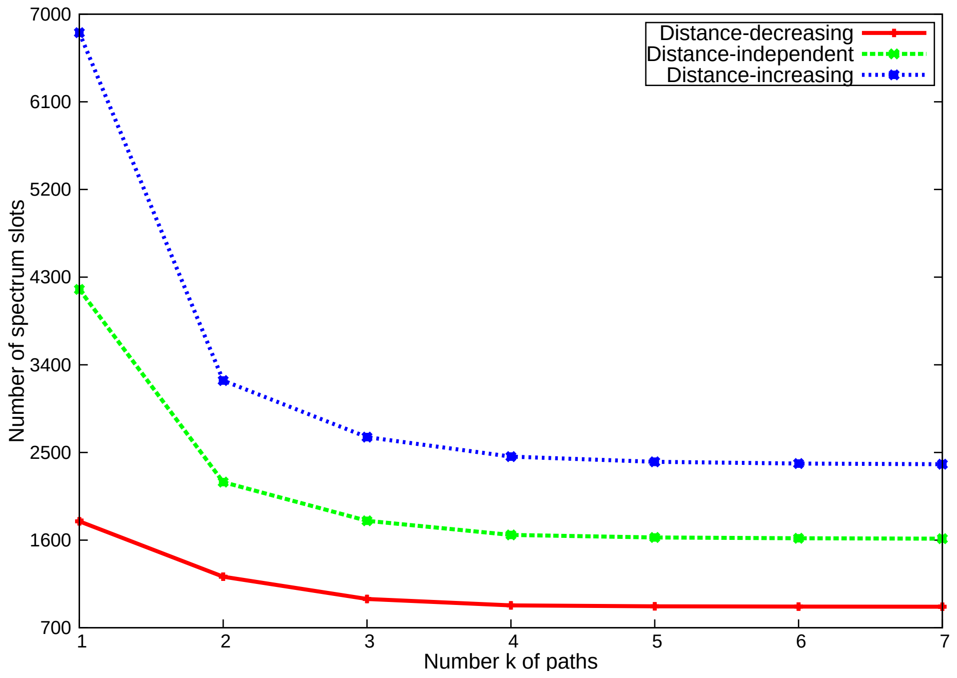


Figure 7.10: Spectrum slots vs. number k of paths in 60-node network

the ratio is due to the increase in the gap between the lower bound and optimal solution as the network size increases. In particular, as the network size grows, spectrum allocation is affected by the interaction of an increasing number of traffic demands over an increasing number of paths and links. Since expression (7.1) for the lower bound does not account for these interactions, we expect that LB becomes looser and further disconnected from the optimal. Therefore, we conjecture that the LS algorithm performs significantly better relative to the lower bound than Figures 7.11-7.13 suggest, especially for the GEANT2 and 60-node networks.

Overall, the results in this section indicate that the LS algorithm is effective in using a small number of alternate paths (i.e., $k = 5, 6$) to utilize spectrum resources efficiently, by balancing the traffic demands across the network links.

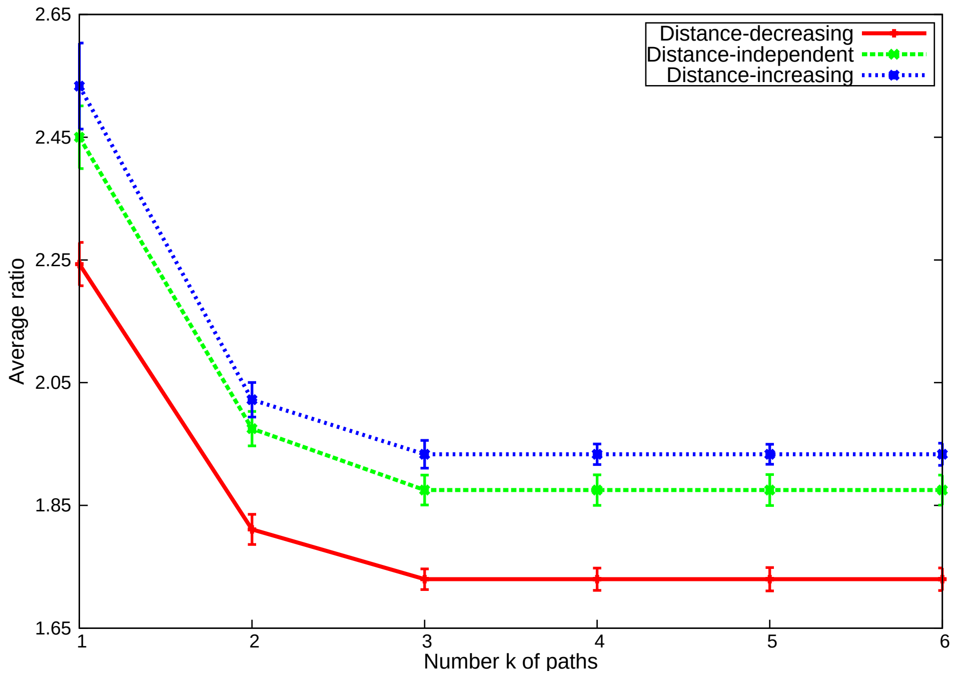


Figure 7.11: Average ratio vs. number k of paths in NSF

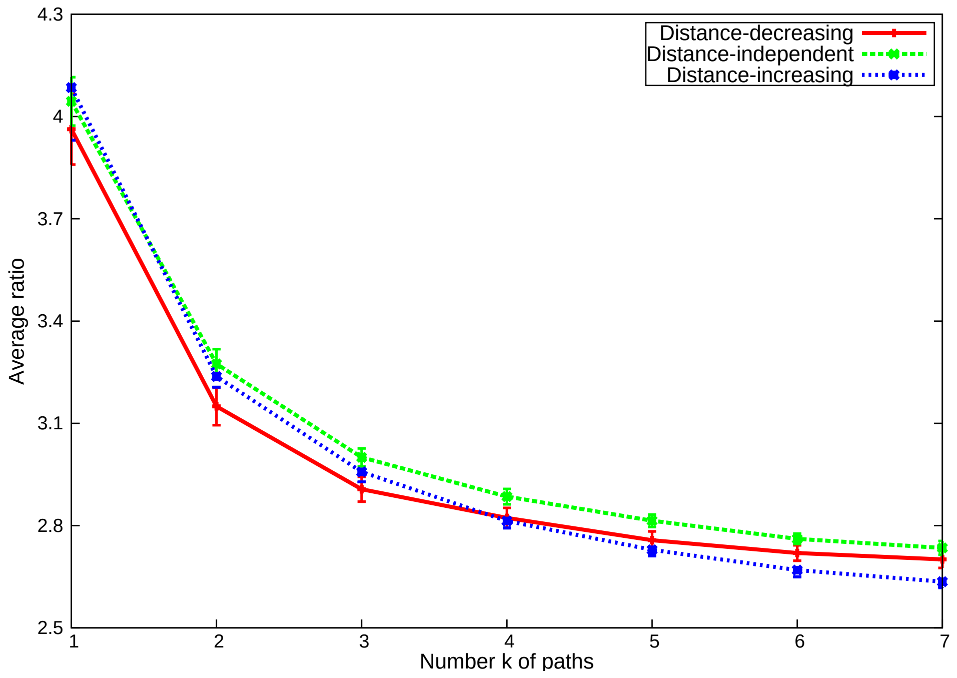


Figure 7.12: Average ratio vs. number k of paths in GEANT2

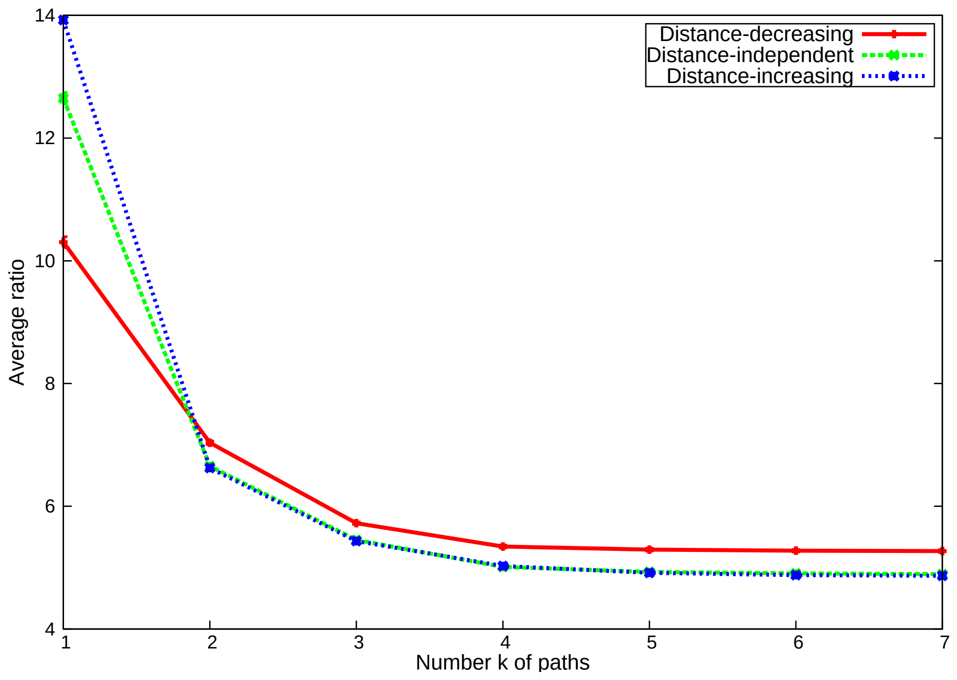


Figure 7.13: Average ratio vs. number k of paths in 60-node network

Chapter 8

Conclusions and Future Work

In this work, we discussed the importance of employing EONs to fulfill the ongoing traffic demands and introduced the constraints that need to be satisfied while we use this type of optical networks. We then reviewed the existing problems in the literature for EONs and classified common solution approaches to tackle each of them. Although there exist various ILP formulations with different assumptions for the RSA problem, they can only handle small network sizes. Hence, the most common approach to solve the RSA problem in the medium to large network instances is (meta)heuristic algorithms.

We then proved that the SA problem in (mesh) networks of general topology is a special case of multiprocessor scheduling problem on the dedicated processors denoted as $Pm|fix_j|C_{max}$. Based on this transformation, we could show that the chain networks with four nodes can be solved in polynomial time, whereas the chain networks with more than five nodes are NP-complete. We also introduced constant-ratio approximation algorithms which beat the previous ratio had been proposed for this problem. Then, we developed four heuristic algorithms inspired by a new scheduling perspective for the SA problem in chain networks and ran extensive simulations to compare their performance with regard to the lower bound for the SA problem.

We also studied the SA problem in rings networks. Concretely, we showed that the SA problem in bidirectional rings with less than four nodes with the shortest-path assumption is solvable in polynomial time. Nevertheless, the SA problem in bidirectional rings with the shortest-path assumption turns out to be NP-complete for rings with more than five nodes. We then developed a 1.5-approximation algorithm for the SA problem in five-node bidirectional rings with the shortest-path routing. Furthermore, we proposed 2-approximation algorithms for the SA problem in six-node and seven-node bidirectional rings under the shortest-path assumption. For rings with more than eight nodes, we developed $3 + \epsilon$-approximation algorithm which is much better than the recently proposed $4 + 2\epsilon$ ratio for SA in rings.

Next, we explored the RSA problem in (mesh) networks of general topology and showed it is a special case of multiprocessor scheduling problem denoted as $Pm|set_j|C_{max}$. Similarly, we proved that the RSA problem for rings with more than four nodes is NP-complete. We could also derive $3+\epsilon$-approximation algorithm for the RSA problem in rings using dynamic programming and an approximation algorithm for the interval chromatic number of interval graphs. Then, we proposed four heuristics algorithms for DA-RSA in ring networks and computed their performance through extensive simulations. These algorithms can be divided in two categories; two of them considers routing and spectrum assignment separately, while the rest considers them jointly.

Finally, we studied the complexity of the RSA problem for general (mesh) networks with different number of nodes. We also developed a fast heuristic algorithm with low computational time to handle DA-RSA in mesh networks. We ran extensive simulations for DA-RSA in NSFNET, GEANT2, and 60-node network and performed statistical analysis to define a confidence interval for the required number of spectrum slots and approximation ratios versus the number of shortest paths in mesh networks.

8.1 Future Work

We proposed a new perspective to solve RSA in elastic optical networks for different topologies. Some of the possible future directions for this problem are summarized as follows.

- **Inverse Multiplexing with Virtual Concatenation in EONs.** Inverse multiplexing with virtual concatenation provides the network with flexibility by splitting traffic demands into multiple streams and sends them independently to the destination. Indeed, virtual concatenation allows better utilization of available spectrum by overcoming the fragmentation which may exist in the EONs. This additional feature simplifies the basic RSA by removing the contiguity constraint. Note when we apply virtual concatenation technique, each split traffic load can only take the value from the set $\{10, 40, 100, 400, 1000\}$ Gbps. For instance, a 100 Gbps traffic demand can be divided into either 10×10 GBps or 2×40 Gbps $+2 \times 10$ Gbps demands. In the multiprocessor scheduling problem perspective, the SA problem with virtual concatenation can transform to the $Pm|fix_j, pmtn|C_{max}$ [102], where *pmtn* stands for *preemption*. Similarly, RSA with virtual concatenation can map to $Pm|set_j, pmtn|C_{max}$ [92] where more than one set can be chosen to perform a task. To the best f our knowledge, there are no results regarding the complexity or approximation algorithm for $Pm|set_j, pmtn|C_{max}$. One possible direction is to develop optimal and approximation algorithms for $Pm|fix_j, pmtn|C_{max}$ and $Pm|set_j, pmtn|C_{max}$ and apply them to solve the RSA problem with virtual concatenation.
- **Survivable EONs.** Survivable elastic optical network is another active area which deals

with intractable ILPs. Another possible future direction would be utilizing the multiprocessor scheduling approaches for tackling the basic RSA and provisioning protection capacity in elastic optical networks. One of extension to our proposed work on RSA would be the RSA with dedicated path protection in which we find two paths (primary and backup) for each connection request. Clearly, the corresponding scheduling problem would be $Pm|set_j|C_{max}$. Specifically, we duplicate each task (i.e., corresponding traffic request) and then solve the scheduling problem such that these duplicated tasks use the disjoint set of processors to be executed. We also could use similar approach to model the RSA problem with shared protection.

- **Online RSA in EONs.** Online RSA, which tries to find a path and spectrum for each upcoming connection request, is the fundamental control problem in EONs. As the connection requests are arriving dynamically, the focus of existing research has been on developing fast (meta)heuristic algorithms. The other future direction would be development of algorithms with competitive ratio to compute how well these algorithm perform; competitive ratio of an algorithm can be defined as the ratio of the performance of the algorithm over the performance of an optimal algorithm which knows the arrival times of all traffic requests in advance. The equivalent multiprocessor scheduling for the online SA and RSA problems can be represented as $Pm|fix_j, r_j|C_{max}$ [88] and $Pm|set_j, r_j|C_{max}$ [103], respectively, where r_j denotes the arrival of the task j. To the best of our knowledge, $Pm|fix_j, r_j|C_{max}$ and $Pm|set_j, r_j|C_{max}$ have not received adequate attention in the literature. Hence, there is potential in working on developing algorithms for these scheduling problems and apply them to solve SA and RSA.

REFERENCES

[1] P. J. Winzer. Beyond 100G Ethernet. *IEEE Communications Magazine*, 48(7):26--30, July 2010.

[2] A. Nag and M. Tornatore. Optical network design with mixed line rates. *Optical Switching and Networking*, 6(3):227--237, 2009.

[3] ITU-T G.694.1. Spectral grids for WDM applications: DWDM frequency grid, February 2002.

[4] G. Shen and M. Zukerman. Spectrum-efficient and agile CO-OFDM optical transport networks: architecture, design, and operation. *IEEE Communications Magazine*, 50(5):82--89, 2012.

[5] M. Jinno, T. Ohara, Y. Sone, A. Hirano, O. Ishida, and M. Tomizawa. Elastic and adaptive optical networks: possible adoption scenarios and future standardization aspects. *IEEE Communications Magazine*, 49(10):164--172, 2011.

[6] M. Jinno, H. Takara, and B. Kozicki. Dynamic optical mesh networks: Drivers, challenges and solutions for the future. In *Proceedings of 35th European Conference on Optical Communication (ECOC)*, page 7.7.4, September 2009.

[7] O. Gerstel, M. Jinno, A. Lord, and S. J B Yoo. Elastic optical networking: a new dawn for the optical layer? *IEEE Communications Magazine*, 50(2):s12--s20, 2012.

[8] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, T. Yoshimatsu, T. Kobayashi, Y. Miyamoto, K. Yonenaga, A. Takada, O. Ishida, and S. Matsuoka. Demonstration of novel spectrum-efficient elastic optical path network with per-channel variable capacity of 40 Gb/s to over 400 Gb/s. In *Proceedings of 34th European Conference on Optical Communication (ECOC)*, page Th.3.F.6, September 2008.

[9] 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, First Quarter 2013.

[10] W. Shieh. OFDM for flexible high-speed optical networks. *Journal of Lightwave Technology*, 29(10):1560--1577, May 15 2011.

[11] A. J. Lowery and J. Armstrong. Orthogonal-frequency-division multiplexing for dispersion compensation of long-haul optical systems. *Optical Express*, 14(6):2079--2084, March 2006.

[12] A. J. Lowery, L. B. Du, and J. Armstrong. Performance of optical OFDM in ultralong-haul WDM lightwave systems. *Journal of Lightwave Technology*, 25(1):131--138, January 2007.

[13] M. Jinno, H. Takara, B. Kozicki, Yukio Tsukishima, Y. Sone, and S. Matsuoka. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *IEEE Communications Magazine*, 47(11):66--73, 2009.

[14] H. Beyranvand and J. Salehi. A quality-of-transmission aware dynamic routing and spectrum assignment scheme for future elastic optical networks. *Journal of Lightwave Technology*, 31(18):3043--3054, September 15 2013.

[15] S. Frisken, G. Baxter, D. Abakoumov, H. Zhou, I. Clarke, and S. Poole. Flexible and grid-less wavelength selective switch using LCOS technology. In *Proceedings of OFC/NFOEC 2011*, page Paper OTuM3, 2011.

[16] R. Ryf, Y. Su, L. Moller, S. Chandrasekhar, X. Liu, D. T. Nelson, and C. R. Giles. Wavelength blocking filter with flexible data rates and channel spacing. *Journal of Lightwave Technology*, 23(1):54--61, January 2005.

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

[18] Y. Wang, X. Cao, and Y. Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proceedings of IEEE INFOCOM*, pages 1503--1511, 2011.

[19] G. N. Rouskas. Routing and wavelength assignment in optical WDM networks. In *J. Proakis (Editor), Wiley Encyclopedia of Telecommunications*. John Wiley & Sons, 2001.

[20] S. Shirazipourazad, Ch. Zhou, Z. Derakhshandeh, and A. Sen. On routing and spectrum allocation in spectrum-sliced optical networks. In *Proceedings of IEEE INFOCOM*, pages 385--389, April 2013.

[21] Y. Wang, X. Cao, and Q. Hu. Routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, June 2011.

[22] Y. Wang, X. Cao, Q. Hu, and Y. Pan. Towards elastic and fine-granular bandwidth allocation in spectrum-sliced optical networks. *Journal of Optical Communications and Networking*, 4(11):906--917, 2012.

[23] A. N. Patel, Ph. N. Ji, J. P. Jue, and T. Wang. Routing, wavelength assignment, and spectrum allocation in transparent flexible optical WDM (FWDM) networks. In *Proceedings of Integrated Photonics Research, Silicon and Nanophotonics and Photonics in Switching (IPR/PS)*, page PDPWG1, July 2010.

[24] A. N. Patel, Ph. N. Ji, J. P. Jue, and T. Wang. Routing, wavelength assignment, and spectrum allocation algorithms in transparent flexible optical WDM networks. *Optical Switching and Networking*, 9(3):191--204, 2012.

[25] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos. Spectrally/bitrate flexible optical network planning. In *Proceedings of 36th European Conference on Optical Communication (ECOC)*, page WE.8.D.3, 2010.

[26] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos. Routing and spectrum allocation in OFDM-based optical networks with elastic bandwidth allocation. In *Proceedings of IEEE Globecom*, 2010.

[27] L. Velasco, M. Klinkowski, M. Ruiz, and J. Comellas. Modeling the routing and spectrum allocation problem for flexgrid optical networks. *Photonic Network Communications*, 24:177--186, 2012.

[28] X. Wan, L. Wang, N. Hua, H. Zhang, and X. Zheng. Dynamic routing and spectrum assignment in flexible optical path networks. In *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, page JWA55, March 2011.

[29] 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 *Proceedings of 37th European Conference on Optical Communication (ECOC)*, page Mo.2.K.6, September 2011.

[30] 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 *Proceedings of OFC/NFOEC 2013*, page Paper JW2A, 2013.

[31] R. C. Almeida Jr., R. A. Delgado, C. J. A. Bastos-Filho, D. A. R. Chaves, H. A. Pereira, and J. F. Martins-Filho. An evolutionary spectrum assignment algorithm for elastic optical networks. In *Proceedings of 15th International Conference on Transparent Optical Networks (ICTON)*, June 2013.

[32] 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 *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, page Paper NTu2J.3, March 2012.

[33] A. N. Patel, Ph. N. Ji, J. P. Jue, and T. Wang. Dynamic routing, wavelength assignment, and spectrum allocation in transparent flexible optical WDM networks. In *Proceedings of SPIE*, volume 7959, pages 79590M--1--79590M--8, January 2011.

[34] A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, Juan Pedro Fernndez-Palacios, and Davide Careglio. Dynamic routing and spectrum (re)allocation in future flexgrid optical networks. *Computer Networks*, 56(12):2869--2883, 2012.

[35] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos. Time-varying spectrum allocation policies and blocking analysis in flexible optical networks. *IEEE Journal on Selected Areas in Communications*, 31(1):13--25, January 2013.

[36] 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.

[37] H. Takara, B. Kozicki, Y. Sone, and M. Jinno. Spectrally-efficient elastic optical path networks. In *Proceedings of 15th OptoeElectronics and Communications Conference (OECC)*, pages 116--117, July 2010.

[38] B. Kozicki, H. Takara, Y. Sone, A. Watanabe, and M. Jinno. Distance-adaptive spectrum allocation in elastic optical path network (SLICE) with bit per symbol adjustment. In *Proceedings of Optical Fiber Communication and National Fiber Optic Engineers Conference (OFC/NFOEC)*, page OMU3, March 2010.

[39] H. Takara, B. Kozicki, Y. Sone, T. Tanaka, A. Watanabe, A. Hirano, K. Yonenaga, and M. Jinno. Distance–adaptive super–wavelength routing in elastic optical path network (SLICE) with optical OFDM. In *Proceedings of 36th European Conference and Exhibition on Optical Communication (ECOC)*, page We.8.D.2, September 2010.

[40] 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 *Proceedings of 37th European Conference on Optical Communication (ECOC)*, page Mo.2.K.3, Sptember 2011.

[41] H. Takara, K. Yonenaga, and M. Jinno. Spectrally-efficient elastic optical path networks toward 1 Tbps era. In *Optical Fiber Communication Conference*, page OTh3B.3, March 2012.

[42] A. Morea and O. Rival. Advantages of elasticity versus fixed data-rate schemes for restorable optical networks. In *Proceedings of 36th European Conference on Optical Communication (ECOC)*, page Th.10.F.5, Sptember 2010.

[43] S. Yang and F. Kuipers. Impairment-aware routing in translucent spectrum-sliced elastic optical path networks. In *Proceedings of 17th European Conference on Networks and Optical Communications (NOC)*, June 2012.

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

[45] T. Takagi, H. Hasegawa, K. 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 *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference(OFC/NFOEC)*, page OTuI7, March 2011.

[46] R. Wang and B. Mukherjee. Spectrum management in heterogeneous bandwidth optical networks. *Optical Switching and Networking*, 11, 2014.

[47] X. Wang, Q. Zhang, I. Kim, P. Palacharla, and M. Sekiya. Utilization entropy for assessing resource fragmentation in optical networks. In *Proceedings of OFC/NFOEC 2012*, page Paper OTh1A.2, 2012.

[48] Z. Zhu, W. Lu, L. Zhang, and N. Ansari. Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing. *Journal of Lightwave Technology*, 31(1):15--22, January 2013.

[49] M. Zhang, W. Lu, Z. Zhu, Y. Yin, and S.J.B. Yoo. Planning and provisioning of elastic O-OFDM networks with fragmentation-aware routing and spectrum assigment (rsa) algorithms. In *Proceedings of Asia Communications and Photonics Conference (ACP)*, volume ATh2D.3, November 2012.

[50] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos. Dynamic bandwidth allocation in flexible OFDM–based networks. In *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, page OTuI5, March 2011.

[51] Y. Sone, A. Hirano, A. Kadohata, M. Jinno, and O. Ishida. Routing and spectrum assignment algorithm maximizes spectrum utilization in optical networks. In *Proceedings of 37th European Conference on Optical Communication (ECOC)*, page Mo.1.K.3, September 2011.

[52] N. Amaya, M. Irfan, G. Zervas, K. Banias, M. Garrich, I. Henning, D. Simeonidou, Y.R. Zhou, A. Lord, K. Smith, V. J F Rancano, S. Liu, P. Petropoulos, and D.J. Richardson. Gridless optical networking field trial: Flexible spectrum switching, defragmentation and transport of 10G/40G/100G/555G over 620-km field fiber. In *Proceedings of 37th European Conference on Optical Communication (ECOC)*, page Th.13.K.1, September 2011.

[53] A.N. Patel, P.N. Ji, J.P. Jue, and T. Wang. Defragmentation of transparent flexible optical WDM (FWDM) networks. In *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference(OFC/NFOEC)*, page OTuI8, March 2011.

[54] K. Wen, Y. Yin, D. J. Geisler, Sh. Chang, and S. J. B. Yoo. Dynamic on-demand lightpath provisioning using spectral defragmentation in flexible bandwidth networks. In *Proceedings of 37th European Conference on Optical Communication (ECOC)*, page Mo.2.K.4, September 2011.

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

[56] 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, January 2013.

[57] R. Dutta, A. E. Kamal, and G. N. Rouskas (Eds.). *Traffic Grooming for Optical Networks: Foundations, Techniques and Frontiers.* Springer, New York, 2008.

[58] R. Dutta and G. N. Rouskas. Traffic grooming in WDM networks: Past and future. *IEEE Network*, 16(6):46--56, November/December 2002.

[59] O. Gerstel. Flexible use of spectrum and photonic grooming. In *Proceedings of Integrated Photonics Research, Silicon and Nanophotonics and Photonics in Switching (IPR/PS)*, volume PMD3, July 2010.

[60] B. Kozicki, H. Takara, Y. Tsukishima, T. Yoshimatsu, T. Kobayashi, K. Yonenaga, and M. Jinno. Optical path aggregation for 1-Tb/s transmission in spectrum-sliced elastic optical path network. *IEEE Photonics Technology Letters*, 22(17):1315--1317, 2010.

[61] G. Zhang, M. De Leenheer, and B. Mukherjee. Optical traffic grooming in OFDM-based elastic optical networks [Invited]. *Journal of Optical Communications and Networking*, 4(11):B17--B25, 2012.

[62] W. Zheng, Y. Jin, W. Sun, W. Guo, and W. Hu. On the spectrum-efficiency of bandwidth-variable optical OFDM transport networks. In *Proceedings of Optical Fiber Communi-*

cation and National Fiber Optic Engineers Conference (OFC/NFOEC), page OWR5, March 2010.

[63] Y. Zhang, X. Zheng, Q. Li, N. Hua, Y. Li, and H. Zhang. Traffic grooming in spectrum-elastic optical path networks. In *Proceedings of Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, page OTuI1, March 2011.

[64] S. Zhang, C. Martel, and B. Mukherjee. Dynamic traffic grooming in elastic optical networks. *IEEE Journal on Selected Areas in Communications*, 31(1):4--12, January 2013.

[65] M. Liu, M. Tornatore, and B. Mukherjee. Survivable traffic grooming in elastic optical networks -- shared protection. *Journal of Lightwave Technology*, 31(6):903--909, March 2013.

[66] D. Zhou and S. Subramaniam. Survivability in optical networks. *IEEE Network*, 14(6):16--23, November/December 2000.

[67] B. Chen, J. Zhang, Y. Zhao, Ch. Lv, W. Zhang, Sh. Huang, X. Zhang, and W. Gu. Multi-link failure restoration with dynamic load balancing in spectrum-elastic optical path networks. *Optical Fiber Technology*, 18(1):21--28, 2012.

[68] T. Takagi, H. Hasegawa, K. Sato, T. Tanaka, B. Kozicki, Y. Sone, and M. Jinno. Algorithms for maximizing spectrum efficiency in elastic optical path networks that adopt distance adaptive modulation. In *Proceedings of 36th European Conference on Optical Communication (ECOC)*, page We.8.D.5, September 2010.

[69] A. Eira, J. Pedro, and J. Pires. Optimized design of shared restoration in flexible-grid transparent optical networks. In *Proceedings of OFC/NFOEC 2012*, page Paper JTh2A37, 2012.

[70] A. N. Patil, P. N. Ji, J. P. Jue, and T. Wang. Survivable transparent flexible optical WDM (FWDM) networks. In *Proceedings of OFC/NFOEC 2011*, page Paper OTuI2, 2011.

[71] X. Shao, Y-K. Yeo, Z. Xu, X. Cheng, and L. Zhou. Shared path protection in OFDM-based optical networks with elastic bandwidth allocation. In *Proceedings of OFC/NFOEC 2012*, page Paper OTh4B.4, 2012.

[72] J. L. Vizcaino, Y. Ye, V. Lopez, F. Jimenez, F. Musumeci, M. Tornatore, A. Pattavina, and P. M. Krummrich. Protection in optical transport networks with fixed and flexible grid: Cost and energy efficiency evaluation. *Optical Switching and Networking*, 11, 2014.

[73] Y. Sone, A. Watanabe, W. Imajuku, Y. Tsukishima, B. Kozicki, H. Takara, and M. Jinno. Highly survivable restoration scheme employing optical bandwidth squeezing in spectrum – sliced elastic optical path (SLICE) network. In *Proceedings of Optical Fiber Communication (OFC)*, page OThO2, March 2009.

[74] Y. Sone, A. Watanabe, W. Imajuku, Y. Tsukishima, B. Kozicki, H. Takara, and M. Jinno. Bandwidth squeezed restoration in spectrum-sliced elastic optical path networks (SLICE). *Journal of Optical Communications and Networking*, 3(3):223--233, 2011.

[75] B. Chen, J. Zhang, Y. Zhao, Ch. Lv, W. Zhang, Y. Gu, Sh. Huang, and W. Gu. A novel recovery algorithm for multi-link failures in spectrum-elastic optical path networks. In *Proceedings of Asia Communications and Photonics Conference (ACP)*, volume 8310, pages 83101V--1--83101V--6, November 2011.

[76] Y. Wei, G. Shen, and Sh. You. Span restoration for CO-OFDM-based elastic optical networks under spectrum conversion. In *Proceedings of Asia Communications and Photonics Conference (ACP)*, page AF3E.7, Novomber 2012.

[77] D. Cavendish, K. Murakami, S-H. Yun, O. Matsuda, and M. Nishihara. New transport services for next-generation SONET/SDH systems. *IEEE Communications Magazine*, 40(5):80--87, May 2002.

[78] E. Hernandez-Valencia. Hybrid transport solutions for tdm/data networking services. *IEEE Communications Magazine*, 40(5):104--112, May 2002.

[79] International Telecommunication Union (ITU). Link capacity adjustment scheme (LCAS) for virtually concatenated signals. In *ITU-T G.7042*, 2004.

[80] X. Chen, Y. Zhong, and A. Jukan. Multi-path routing in elastic optical networks with distance-adaptive modulation formats. In *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2013.

[81] W. Lu, X. Zhou, L. Gong, M. Zhang, and Z. Zhu. Dynamic multi-path service provisioning under differential delay constraint in elastic optical networks. *IEEE Communications Letters*, 17(1):158--160, January 2013.

[82] X. Chen, A. Jukan, and A. Gumaste. Optimized parallel transmission in elastic optical networks to support high-speed Ethernet. *Journal of Lightwave Technology*, 32(2):228--238, January 15 2014.

[83] A. Castro, L. Velasco, M. Ruiz, and J. Comellas. Single-path provisioning with multi-path recovery in flexgrid optical networks. In *Proceedings of the 4th International Workshop on Reliable Networks Design and Modeling (RNDM)*, 2012.

[84] L. Ruan and N. Xiao. Survivable multipath routing and spectrum allocation in OFDM-based flexible optical networks. *Journal of Optical Communications and Networking*, 5(3):172--182, March 2013.

[85] R. Bhandari. *Survivable Networks: Algorithms for Diverse Routing.* Kluwer, USA, 1999.

[86] L. Ruan and Y. Zheng. Dynamic survivable multipath routing and spectrum allocation in OFDM-based flexible optical networks. *Journal of Optical Communications and Networking*, 6(1):77--85, January 2014.

[87] E. Bampis, M. Caramia, J. Fiala, A. Fishkin, and A. Iovanella. Scheduling of independent dedicated multiprocessor tasks. In *Proceedings of the 13th Annual International Symposium on Algorithms and Computation*, volume LNCS 2518, pages 391--402, 2002.

[88] J. A. Hoogeveen, S. L. Van de Velde, and B. Veltman. Complexity of scheduling multiprocessor tasks with prespecified processor allocations. *Discree Applied Mathematics*, 55:259--272, 1994.

[89] E. Bampis and A. Kononov. On the approximability of scheduling multiprocessor tasks with time dependent processing and processor requirements. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, San Francisco, 2001.

[90] L. Bianco, J. Blazewicz, P. Dell'Olmo, and M. Drozdowski. Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors. *Annals of Operations Research*, 58(7):493--517, 1995.

[91] J. Chen and Ch. Lee. General multiprocessor task scheduling. *Naval Research Logistics*, 46(1):57--74, 1999.

[92] K. Jansen and L. Porkolab. General multiprocessor task scheduling: Approximate solutions in linear time. *SIAM Journal on Computing*, 35(3):519--530, 2005.

[93] L. Torres A. Miranda and J. Chen. On the approximability of multiprocessor task scheduling problems. In *Proceedings of the 13th Annual International Symposium on Algorithms and Computation*, volume LNCS 2518, pages 403--415, 2002.

[94] M. Kubal. The complexity of scheduling independent two-processor tasks on dedicated processors. *Information Processing Letters*, 24(3):141--147, 1987.

[95] J. Chen and A. Miranda. A polynomial time approximation scheme for general multiprocessor job scheduling. *SIAM Journal on Computing*, 31(1):1--17, 2001.

[96] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., New York, 1979.

[97] H. A. Kierstead. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics*, 88(23):231--237, 1991.

[98] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. Opt versus load in dynamic storage allocation. *SIAM Journal of Computing*, 33(3):632--646, 2004.

[99] J. Huang, J. Chen, S. Chen, and J. Wang. A simple linear time approximation algorithm for multi-processor job scheduling on four processors. *Journal of Combinatorial Optimization*, 13:33--45, 2007.

[100] Darpa core optical networks (CORONET) continental united states (CONUS) topology. http://monarchna.com/CORONET_CONUS_Topology.xls.

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

[102] A. K. Amoura, E. Bampis, C. Kenyon, and Y. Manoussakis. Scheduling independent multiprocessor tasks. In *Proceedings of ESA '97*, volume LNCS 1284, pages 1--12, 1997.

[103] L. Bianco, J. Blazewicz, P. Dell'Olmo, and M. Drozdowski. Spreemptive multiprocessor task scheduling with release times and time windows. *Annals of Operations Research*, 70(1-4):43--55, 1997.

APPENDICES

Appendix A

$P4|line_j|C_{max}$ Approximation Ratio

We show that there exists an approximation algorithm with ratio of $1.5 - \epsilon$ where $\epsilon > 0$ for $P4|line_j|C_{max}$. We assume that the subscript of each task denotes the set of processors required to execute the task. For the sake of simplicity of the proof, it is assumed that the processing time of T_{1234} is zero. This assumption does not impact the proof, in that T_{1234} can be scheduled first. We also use L_i notation to represent total processing time on processor i, for $i = 1, \ldots, 4$. Let LB also presents the lower bound and OPT be the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

The main idea behind this algorithm is to divide all the tasks requiring the busiest processor into two disjoint sets and schedule the more weighted one first, then try to execute the remained tasks to the end of this schedule. In case that the constructed gaps in the proposed schedule do not provide enough space for any of the remained tasks, we use *dragging technique*. This technique allows us to expand theses gaps for single-processor tasks until they fit into them. As this problem is symmetric, we only discuss the cases in which processor 1 and 2 have the maximum load and the results can easily be generalized to the cases where processors 3 and 4 are the busiest ones.

A.1 $LB = L_1$

Divide all the tasks requiring processor 1 (i.e., T_1, T_{12}, and T_{123}) into two disjoint sets such that set one contains T_1 and T_{12}, and set two includes T_{123}. Define $\mathcal{C}_1 = T_1 + T_{12}$ and $\mathcal{C}_2 = T_{123}$. We investigate two possibilities which may happen between $\mathcal{C}_1$ and $\mathcal{C}_2$. It is obvious that $\mathcal{C}_1 \geq \mathcal{C}_2$ denotes $\mathcal{C}_1 \geq 0.5\, L_1 = 0.5\, LB$ and vice versa.

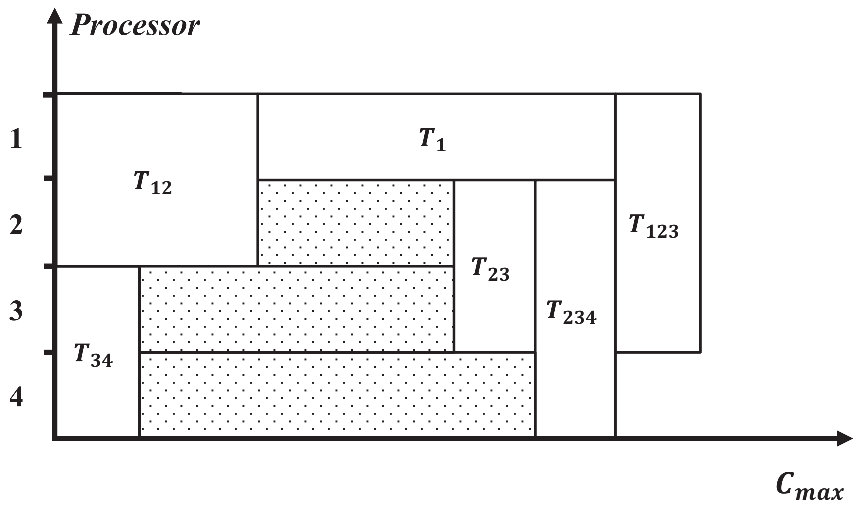


Figure A.1: The schedule corresponding with $LB = L_1$ and $\mathcal{C}_1 \geq 0.5\,LB$

A.1.1 $\mathcal{C}_1 \geq 0.5\,LB$

A schedule for this case is represented in Figure A.1. From assumption $\mathcal{C}_1 \geq 0.5\,LB$, we can infer $\mathcal{C}_2 < 0.5\,LB$. We use this result to show that C_{max} can be bounded. Clearly, T_2 and T_3 can fit into the shaded areas without any issue. If T_4 is small enough to fit into the provided space, then the proposed schedule is the optimum one with $C_{max} = LB$. If not, the dragging technique makes enough room for T_4 and the makespan is modified as $C_{max} = L_4 + T_{123}$. Therefore, we get following set of inequalities

$$\begin{aligned} C_{max} &= L_4 + T_{123} < L_4 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{A.1}$$

A.1.2 $\mathcal{C}_1 < 0.5\,LB$

Similarly, we propose a schedule for this case as illustrated in Figure A.2. Based on the proposed schedule, the shaded spaces are big enough for T_2 and T_4. Still, T_3 may face a problem to fit into its shaded gap. If so, the dragging technique builds enough room for T_3 and makespan is

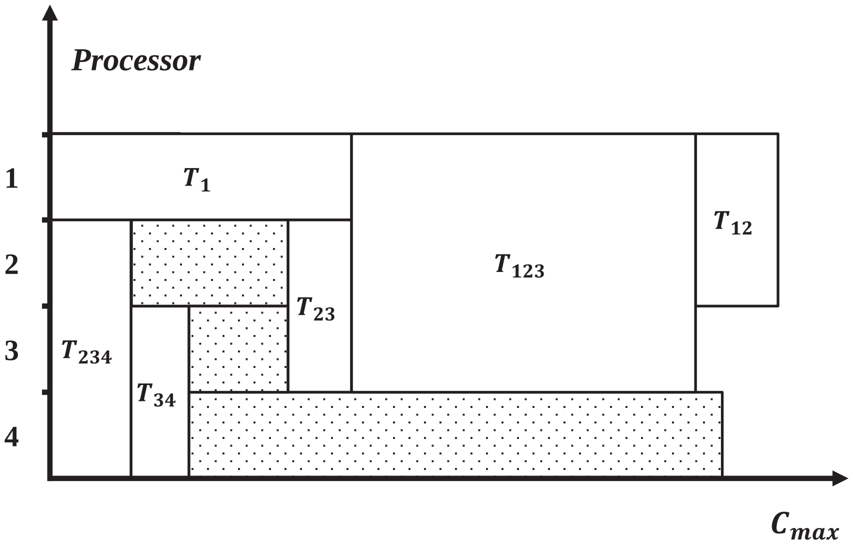


Figure A.2: The schedule corresponding with $LB = L_1$ and $\mathcal{C}_1 < 0.5\,LB$

updated as $C_{max} = L_3 + T_{12}$, and we get the following bound

$$\begin{aligned} C_{max} &= L_3 + T_{12} < L_3 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{A.2}$$

A.2 $LB = L_2$

We follow the same lines of reasoning for the case in which L_2 has the maximum load. We divide the tasks that require processors 2 into two sets such that set one includes T_{12} and T_{123} and set two contains T_2, T_{23} and T_{234}. Define $\mathcal{C}'_1 = T_{12} + T_{123}$ and $\mathcal{C}'_2 = T_2 + T_{23} + T_{234}$. We prove that under either $\mathcal{C}'_1 \geq \mathcal{C}'_2$ or $\mathcal{C}'_1 < \mathcal{C}'_2$ constraint, the completion time of the proposed schedule is strictly less than $1.5 \times OPT$.

A.2.1 $\mathcal{C}'_1 \geq 0.5\,LB$

Build a schedule as illustrated in Figure A.3. Clearly, the assumption here denotes that $T_{23} < 0.5\,LB$, in that $\mathcal{C}'_1 = T_{12} + T_{123} \geq 0.5\,LB$. This result helps us to define a bound for the makespan. There would be no problem to assign all the remained tasks except T_4. If T_4 is small enough to fit in the shaded area, then the represented schedule in Figure A.3 is optimal (i.e. $C_{max} = LB$). If not, then the dragging technique is called to make enough space and the makespan is computed

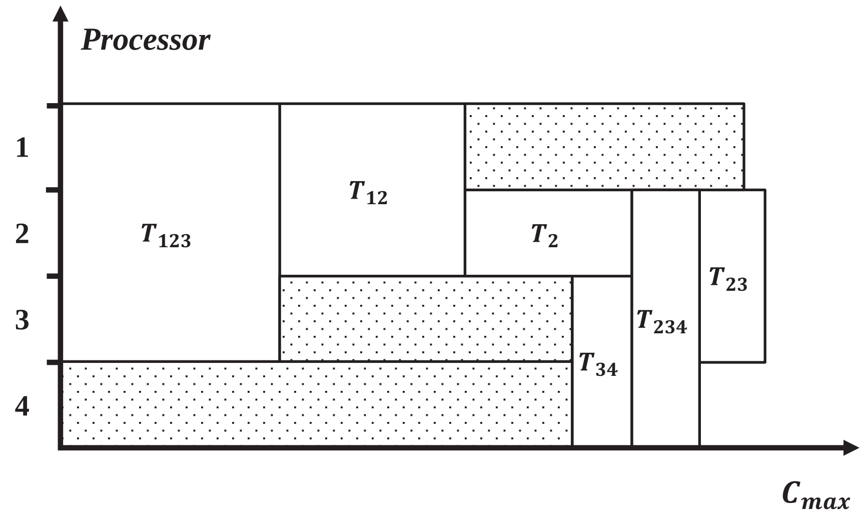


Figure A.3: The schedule corresponding with $LB = L_2$ and $\mathcal{C}'_1 \geq 0.5\,LB$

as $C_{max} = L_4 + T_{23}$. Therefore, we could derive an upper bound for C_{max} in the worst case as following

$$\begin{aligned} C_{max} &= L_4 + T_{23} < L_4 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{A.3}$$

A.2.2 $\mathcal{C}'_1 < 0.5\,LB$

This case is the reverse of previous assumption. The schedule of the tasks under this scenario is shown in Figure A.4. The order of the tasks assures that T_1 and T_4 get enough space. If T_3 fits, the schedule is optimal; otherwise, the dragging technique yields a schedule with completion time defined as $C_{max} = L_3 + T_{12}$ and the upper bound for C_{max} can be summarized as following

$$\begin{aligned} C_{max} &= L_3 + T_{12} < L_3 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{A.4}$$

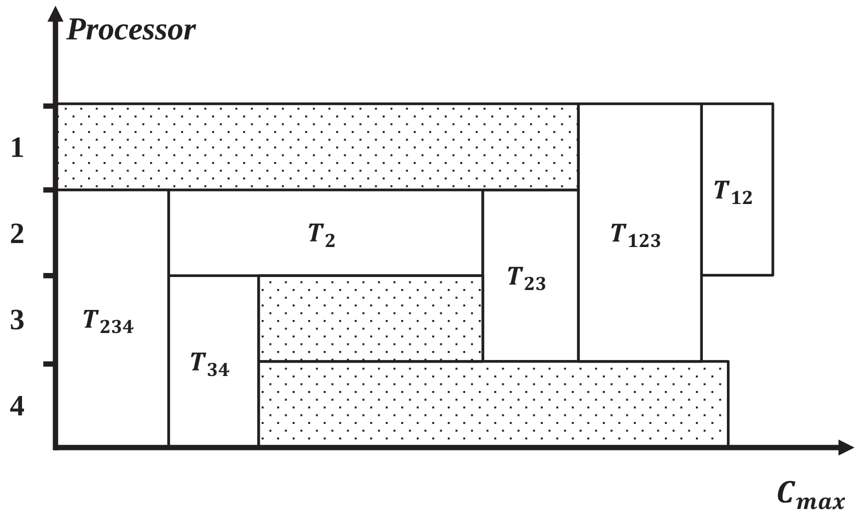


Figure A.4: The schedule corresponding with $LB = L_2$ and $\mathcal{C}_1' < 0.5\,LB$

Appendix B

$P5|line_j|C_{max}$ Approximation Ratio

Similar to our discussion in Appendix A, we show that there exists a $(1.5-\epsilon)$-approximation algorithm for $P5|line_j|C_{max}$ scheduling problem where $\epsilon > 0$. Again, we assume that the subscript of each task denotes the set of processors required to execute the task. It is also assumed that the processing time of T_{12345} is zero. This assumption does not affect the proof as this task can be scheduled first. We use L_i to denote the total processing time on processor i, for $i = 1, \ldots, 5$. Let LB also presents the lower bound and OPT be the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

The idea is similar to Appendix A; that is, all the tasks on the busiest processor are split into two sets. Then, the set with greater total processing time is scheduled first. Finally, the remained tasks are added based on prespecified order to the schedule such that the tasks with shorter processing times come at the end of the schedule. Similarly, whenever the provided space is not enough for some task, we use the *dragging technique* to make enough space for it. As this problem is symmetry, we only need to discuss the cases where the LB happens at either processor 1, 2, or 3. This approach can be generalized to the case where either processor 4 or 5 is the processor with the maximum load.

B.1 $LB = L_1$

Consider the set of tasks requiring processor 1 (i.e., T_1, T_{12}, T_{123}, and T_{1234}). We prove that we can split these tasks into two disjoint sets and construct a schedule with ratio of 1.5 such that set one contains only T_1 and T_{12} and set two includes T_{123} and T_{1234}. Let $\mathcal{C}_1 = T_1 + T_{12}$ and $\mathcal{C}_2 = T_{123} + T_{1234}$. For the rest of our discussion, we compare the total processing times of these two sets. If $\mathcal{C}_1 = T_1 + T_{12} \geq T_{123} + T_{1234} = \mathcal{C}_2$, then we could conclude that $\mathcal{C}_1 \geq 0.5\, L_1 = 0.5\, LB$. We use the latter inequality for the sake of simplicity.

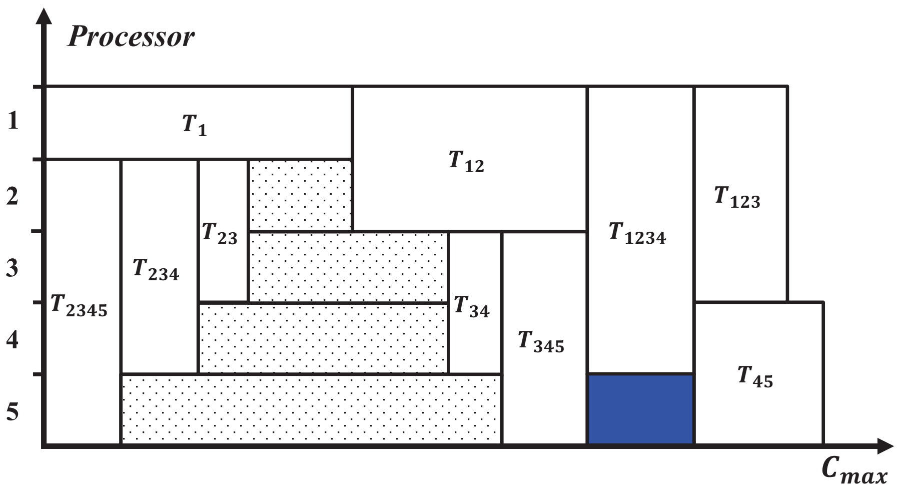


Figure B.1: The schedule corresponding with $LB = L_1$, $\mathcal{C}_1 \geq 0.5\,LB$, and $T_{45} < 0.5\,LB$

B.1.1 $\mathcal{C}_1 \geq 0.5\,LB$

The assumption here implies that $\mathcal{C}_2 = T_{123} + T_{1234} < 0.5\,LB$ and the order of tasks are shown in Figure B.1. We consider two possible scenarios between T_{123} and T_{45} values (i.e., either $T_{123} \geq T_{45}$ or $T_{123} < T_{45}$) and show the proposed schedule is always less than $1.5 \times OPT$.

B.1.1.1 $T_{123} \geq T_{45}$

As it is shown in Figure B.1, T_4 or T_5 might not fit into the shaded areas. We study each case separately and calculate the worst possible C_{max} as following

T_4 **case:** The dragging technique leads to a schedule with a new completion time modified as $C_{max} = L_4 + pace$ where $pace \leq T_{123} - T_{45}$ as the total processing on processor 4 cannot exceed the total processing time on processor 1. Based on this observation, the upper bound for the C_{max} is estimated as

$$\begin{aligned} C_{max} &= L_4 + pace \leq L_4 + T_{123} - T_{45} \leq L_4 + T_{123} \\ &< L_4 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.1}$$

T_5 **case:** The updated completion time would be $C_{max} = L_5 + pace$ where $pace \leq T_{1234} + (T_{123} - T_{45})$; otherwise, the L_5 is greater than LB. Hence, an upper bound for the makespan in this case would be stated as

$$\begin{aligned} C_{max} &= L_5 + pace \leq L_5 + T_{1234} + (T_{123} - T_{45}) \leq L_5 + T_{1234} + T_{123} \\ &< L_5 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.2}$$

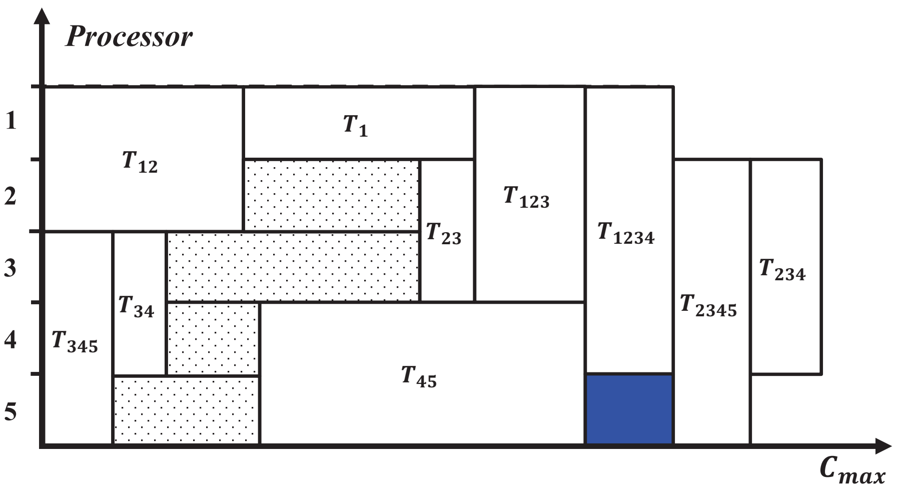


Figure B.2: The schedule corresponding with $LB = L_1$, $\mathcal{C}_1 \geq 0.5\,LB$, and $T_{45} \geq 0.5\,LB$

Thus, we can conclude that under $T_{123} \geq T_{45}$ constraint, the purposed schedule in Figure B.1 has $(1.5 - \epsilon)$-approximation ratio where $\epsilon > 0$.

B.1.1.2 $T_{123} < T_{45}$

The schedule in Figure B.1 shows that only T_5 may not fit in the designed area. If so, the C_{max} is updated as $C_{max} = L_5 + pace$ such that $pace \leq T_{1234} - (T_{45} - T_{123})$. The upper bound for the $pace$ value guarantees that the L_5 is always less than or equal to the L_1. Subsequently, the following inequality is derived

$$\begin{aligned} C_{max} &= L_5 + pace \leq L_5 + T_{1234} - (T_{45} - T_{123}) \leq L_5 + T_{1234} + T_{123} \\ &< L_5 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.3}$$

However, if the current schedule does not require us to use the dragging technique, the completion time would be calculated as $C_{max} = L_1 + T_{45} - T_{123}$. If $T_{45} < 0.5 \times LB$, then we easily derive 1.5 approximation ratio; otherwise, we construct a new schedule shown in Figure B.2. If we use the same lines of reasoning, we could show that the completion time of this new schedule can still be bounded by $1.5 \times OPT$.

B.1.2 $\mathcal{C}_1 < 0.5\,LB$

Figure B.3 represents a schedule of tasks for this case. This schedule provides adequate space for T_2, T_3, and T_4 where the makespan is defined as $C_{max} = L_1 + T_{2345} + T_{345} + T_{34}$. We also

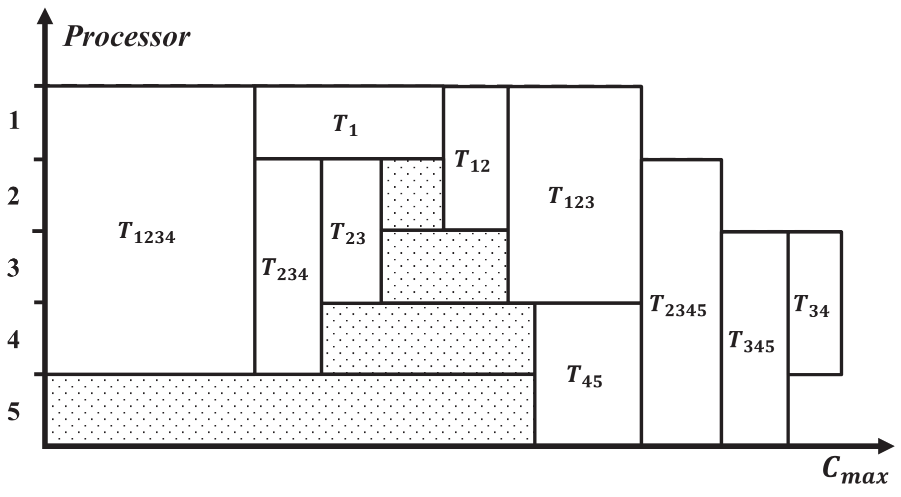


Figure B.3: The schedule corresponding with $LB = L_1$ and $\mathcal{C}_1 < 0.5\,LB$

can derive the following inequality

$$L_3 < L_1 \;\Rightarrow\; T_{34} + T_{345} + T_{2345} \leq T_1 + T_{12} \tag{B.4}$$

In Figure B.3, we observe that if T_5 fits into its shaded area, then the upper bound for the makespan with the use of inequality B.4 can be established as following

$$\begin{aligned} C_{max} &= L_1 + T_{2345} + T_{345} + T_{34} \leq L_1 + T_1 + T_{12} \\ &< L_1 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.5}$$

Otherwise, the dragging technique results in a schedule with the completion time of $C_{max} = L_5 + T_{34}$. Based on the upper bound we obtained for T_{34} in inequality B.4, we conclude that the possible worst value for C_{max} can be computed as

$$\begin{aligned} C_{max} &= L_5 + T_{34} \leq L_5 + T_1 + T_{12} \\ &< L_5 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.6}$$

B.2 $LB = L_2$

Similarly, consider all the tasks requiring processor 2 and divide them into two sets; set one contains T_{23}, T_{123}, and T_{1234}, and set two includes T_2, T_{12}, T_{234}, and T_{2345}. Define $\mathcal{C}'_1 = T_{23} + T_{123} + T_{1234}$ and $\mathcal{C}'_2 = T_2 + T_{12} + T_{234} + T_{2345}$. If $\mathcal{C}'_1 = T_{23} + T_{123} + T_{1234} \geq T_2 + T_{12} + T_{234} + T_{2345} = \mathcal{C}'_2$, we can infer that $\mathcal{C}'_1 \geq 0.5\,LB$.

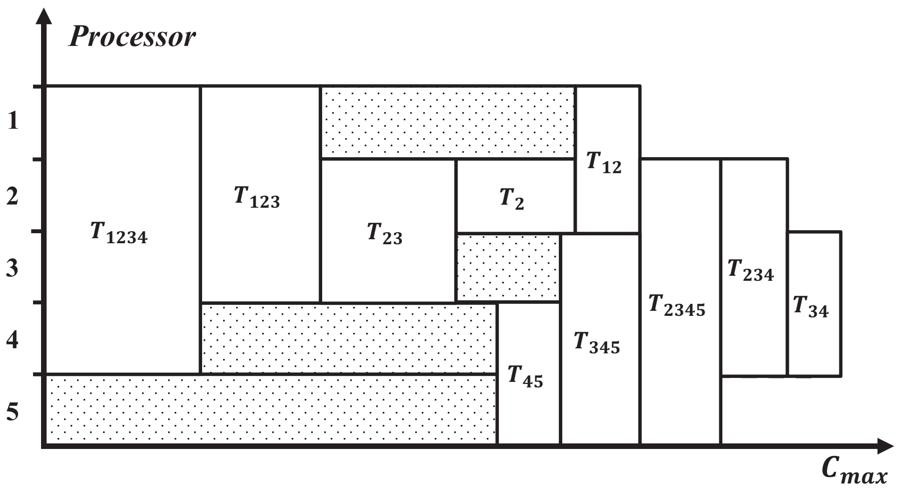


Figure B.4: The schedule corresponding with $LB = L_2$ and $\mathcal{C}'_1 \geq 0.5\,LB$

B.2.1 $\mathcal{C}'_1 \geq 0.5\,LB$

This assumption implies that $\mathcal{C}'_2 = T_2 + T_{12} + T_{234} + T_{2345} < 0.5\,LB$. We use this constraint for the rest of our discussion in this section. A schedule corresponding to this assumption is illustrated in Figure B.4 and it grantees T_3 and T_4 have enough space to be excuted. If T_1 and T_5 also fit into the provided gaps, the makespan corresponds to this schedule is defined as $C_{max} = L_2 + T_{34}$ where T_{34} can be bounded as

$$L_3 < L_2 \;\Rightarrow\; T_3 + T_{34} < T_2 + T_{12} \tag{B.7}$$

If both T_1 and T_5 fit into the provided space, then the completion time of the schedule with the help of derived inequality B.7 and the upper bound value for $\mathcal{C}'_2$ will be determined as

$$\begin{aligned} C_{max} &= L_2 + T_{34} \leq L_2 + T_2 + T_{12} \\ &< L_2 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.8}$$

Now, suppose that any of tasks T_1 or T_5 do not fit in the constructed gaps in the schedule represented in Figure B.4. In this case, the dragging technique is called to expand these gaps. In the following, we address each case separately and show that C_{max} is still bounded by 1.5.

T_1 **case:** The dragging technique will update the $C_{max} = L_1 + T_{2345} + T_{234} + T_{34}$. Using inequality B.7, the makespan can be bounded as

$$\begin{aligned} C_{max} &= L_1 + T_{2345} + T_{234} + T_{34} < L_1 + T_{2345} + T_{234} + (T_2 + T_{12}) \\ &< L_1 + 0.5 \times LB \leq 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.9}$$

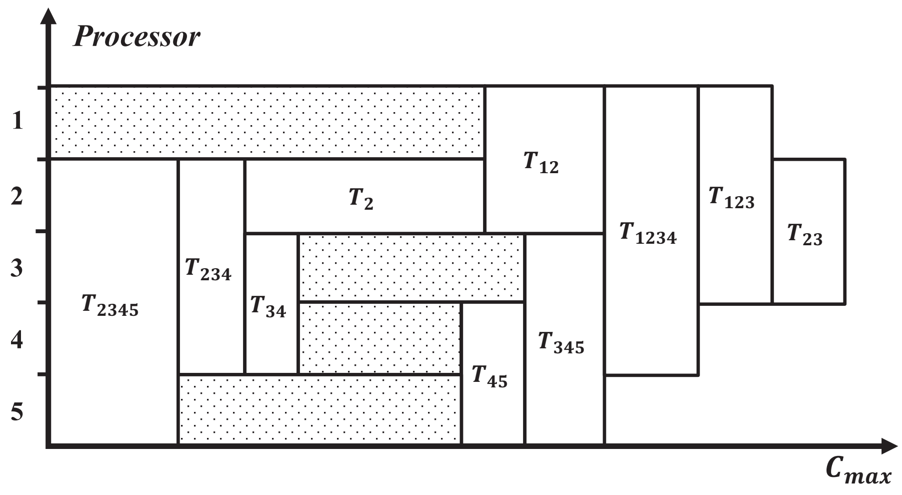


Figure B.5: The schedule corresponding with $LB = L_2$ and $\mathcal{C}_1' < 0.5\,LB$

T_5 **case:** The new completion time after using dragging technique is $C_{max} = L_5 + T_{234} + T_{34}$ and it can be similarly estimated as following

$$\begin{aligned} C_{max} &= L_5 + T_{234} + T_{34} < L_5 + T_{234} + (T_2 + T_{12}) \\ &< L_5 + 0.5 \times LB < 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \quad \text{(B.10)}$$

With respect to these two bounds for the makespan, it can be derived that if $LB = L_2$, the approximation ratio is still strictly less than 1.5.

B.2.2 $\mathcal{C}_1' < 0.5\,LB$

Under this assumption, a new schedule of the tasks is proposed in Figure B.5. We use the same lines of reasoning to show that the approximation ratio is still strictly less than 1.5. As it is illustrated in Figure B.5, T_3 has enough space to fit in, but T_1, T_4, and T_5 are the tasks which may cause problems when they want to be placed in their gaps. Hence, we evaluate the worst case scenario which may happen for any of them as following

T_1 **case:** By using the dragging technique, the makespan of the schedule is defined as $C_{max} = L_1 + T_{23}$. On the other hand, the assumption here implies that $T_{23} < 0.5\,LB$, in that $\mathcal{C}_1' < 0.5\,LB$. This inequality will help us to define the upper bound for C_{max}

$$\begin{aligned} C_{max} &= L_1 + T_{23} < L_1 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \quad \text{(B.11)}$$

T_4 **case:** If T_4 requires us to apply the dragging technique, the makespan of the schedule is updated as $C_{max} = L_4 + T_{123} + T_{23}$. The assumption for $\mathcal{C}_1'$ upper bound leads us to derive the following inequality

$$\begin{aligned} C_{max} &= L_4 + T_{123} + T_{23} < L_4 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.12}$$

T_5 **case:** The last case deals with the case when T_5 does not fit into its designed space. Hence, C_{max} here is estimated as $C_{max} = L_5 + T_{1234} + T_{123} + T_{23}$ and the bound of the makespan can be computed as

$$\begin{aligned} C_{max} &= L_5 + T_{1234} + T_{123} + T_{23} < L_5 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.13}$$

All these two scenarios complete our proof to show that if processor 2 has the maximum load, there is an approximation algorithm to schedule the tasks with ratio strictly less than 1.5.

B.3 $LB = L_3$

Similar to our approach in sections B.1 and B.2, we study the case in which processor 3 has the maximum load and split it into two subproblems. Suppose $\mathcal{C}_1'' = T_{23} + T_{123} + T_{1234}$ and $\mathcal{C}_2'' = T_3 + T_{34} + T_{234} + T_{345} + T_{2345}$. We then investigate the cases when either $\mathcal{C}_1'' \geq 0.5\,LB$ or $\mathcal{C}_1'' < 0.5\,LB$ is valid and discuss each of them separately.

B.3.1 $\mathcal{C}_1'' \geq 0.5\,LB$

Under this assumption, build a schedule similar to Figure B.6. We prove that this order of tasks gives us a ratio less than 1.5. Based on Figure B.6, only T_1, T_2, or T_5 may not fit into the shaded areas. To resolve this problem, the dragging technique is called to make adequate space. From the assumption here, it can be inferred $\mathcal{C}_2'' = T_3 + T_{34} + T_{234} + T_{345} + T_{2345} < 0.5\,LB$. We use this constraint to define an upper bound for the completion time of the proposed schedule.

T_1 **case:** After applying the dragging technique, the new completion time of the proposed schedule as shown in Figure B.6 is estimated as $C_{max} = L_1 + T_{2345} + T_{345} + T_{34}$. With respect to $\mathcal{C}_2''$ upper bound here, we compute C_{max} bound as following

$$\begin{aligned} C_{max} &= L_1 + T_{2345} + T_{345} + T_{34} < L_1 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.14}$$

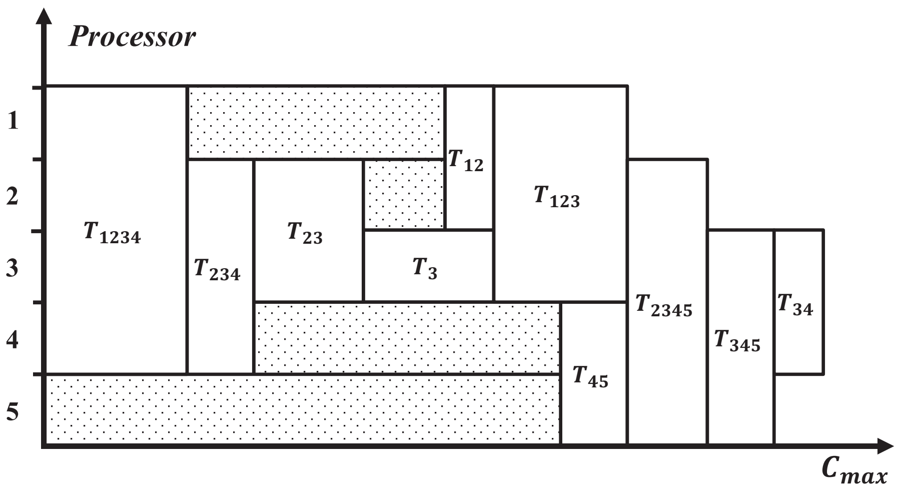


Figure B.6: The schedule corresponding with $LB = L_3$ and $\mathcal{C}_1'' \geq 0.5\,LB$

T_2 **case:** Similar to the previous case, the modified makespan will be defined as $C_{max} = L_2 + T_{345} + T_{34}$. Similarly, we derive the following bounds for the makespan.

$$\begin{aligned} C_{max} &= L_2 + T_{345} + T_{34} < L_2 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.15}$$

T_5 **case:** The new completion time is $C_{max} = L_5 + T_{34}$. Thus, we derive the upper bound for this new makespan as following

$$\begin{aligned} C_{max} &= L_5 + T_{34} < L_5 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.16}$$

All of these discussed cases show that if $LB = L_3$ and $\mathcal{C}_1'' \geq 0.5\,LB$, then there is a schedule with C_{max} bounded by $1.5 \times OPT$.

B.3.2 $\mathcal{C}_1'' < 0.5\,LB$

We continue our the discussion for this case and propose a new schedule of the tasks as represented in Figure B.7. If T_1, T_4, and T_5 could fit into provided gaps, then this schedule is optimum and the approximation ratio is one; otherwise, the dragging technique is called to make more space for these single-processor tasks.

T_1 **case:** The modified makespan is defined as $C_{max} = L_1 + T_{23}$. On the other hand, the

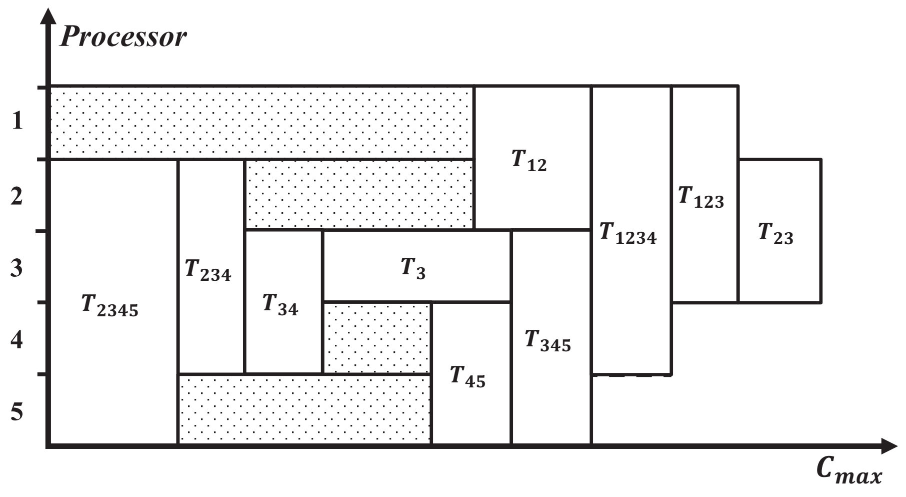


Figure B.7: The schedule corresponding with $LB = L_3$ and $\mathcal{C}_1'' < 0.5\, LB$

assumption here implies that $T_{23} < 0.5\, LB$, so we get the following bound for C_{max}

$$\begin{aligned} C_{max} &= L_1 + T_{23} < L_1 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.17}$$

T_4 **case:** The completion time for this case would be estimated as $C_{max} = L_4 + T_{123} + T_{23}$. We bound the makespan as follwing

$$\begin{aligned} C_{max} &= L_4 + T_{123} + T_{23} < L_4 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.18}$$

T_5 **case:** Based on Figure B.7, the makespan is calculated $C_{max} = L_5 + T_{1234} + T_{123} + T_{23}$ and we derive its bound as following

$$\begin{aligned} C_{max} &= L_5 + T_{1234} + T_{123} + T_{23} < L_5 + 0.5 \times LB \\ &< 1.5 \times LB \leq 1.5 \times OPT \end{aligned} \tag{B.19}$$

Considering our discussion for these three cases, we derive that under $\mathcal{C}_1'' < 0.5\, LB$ assumption, we can build a schedule which is always bounded by $1.5 \times OPT$.

Appendix C

$N = 6, 7$ Nodes Ring with Shortest Path Routing Approximation Ratios

Similar to the discussion in the proof of Lemma 5.1.4, we use constructive approach to show that there exist 2-approximation algorithms for the $N = 6, 7$ nodes ring with shortest path routing. In Section C.1, we provide the argument for the accuracy of this statement in 6-node bidirectional rings under shortest path assumption. Then, we use similar approach to show in Section C.2 that it is also possible to obtain 2-approximation ratio in 7-node bidirectional rings and shortest path routing. Here, we assume that the subscript of each task denotes the set of processors required to execute the task.

C.1 6-Node Ring and Shortest Path Routing

Since 6-node rings have even number of nodes, the clockwise and counterclockwise paths between two symmetric nodes are of equal length (i.e., three), and either may be selected as the shortest path. Let us consider the case where all demands between symmetric nodes are routed in the clockwise direction. That is, we only use the links on the clockwise direction to carry over the traffic demands from node 1 to node 4 and vice versa.

Clearly, the input to the SA subproblem consists of six one-link demands, six two-link demands, and six three-link demands. Hence, the clockwise direction has to serve 18 demands which means that the corresponding scheduling problem contains 18 tasks. Without loss of generality, let processor 3 be the dominant processor, i.e., the one that achieves the lower bound LB in (5.2). Let OPT also represent the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

Consider now a set of nine tasks that do *not* require processor 3 and 4 and remove task T_{61} from this set. Then, schedule these tasks as shown in left part of Figure C.1 with eight tasks

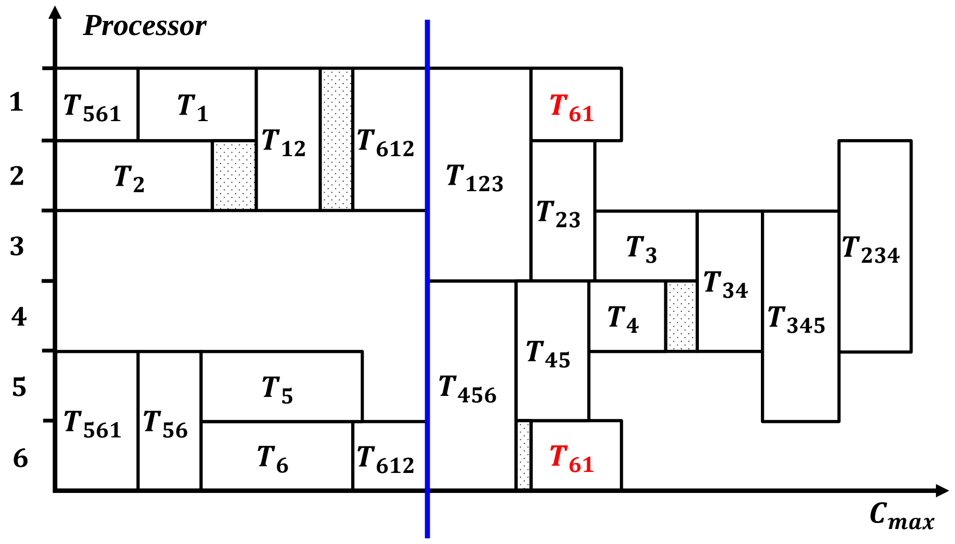


Figure C.1: A six-node bidirectional ring with shortest path routing

needed to be scheduled on four-processor system (i.e., one without processors 3 and 4). It can be easily seen that the proposed schedule in the first part of Figure C.1 is optimum. Let OPT' denote the makespan of this schedule; then, $OPT' \leq OPT$.

In the next step, consider the nine remained tasks that require either processors 3 or 4. These tasks can be scheduled back-to-back without any gaps, as shown in the right part of Figure C.1, whereas T_{61} can be scheduled in parallel with them without no restriction. The makespan corresponding to the second part of the schedule is equal to LB. So, the total makespan of the two-part, 18-task schedule depicted in Figure C.1, is equal to: $OPT' + LB \leq 2 \times OPT$.

C.2 7-Node Ring and Shortest Path Routing

Since the number of nodes is odd, there is only a unique shortest path for every traffic between each pair of nodes. Consequently, the clockwise direction has to serve $21 (= 7 * 6/2)$ demands which means that the corresponding scheduling problem contains 21 tasks. Without loss of generality, let processor 4 be the dominant processor, i.e., the one that achieves the lower bound LB in (5.2). Let OPT also represent the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

Now, consider a set of fifteen tasks that do *not* require processor 4 and remove T_{712}, T_{671}, and T_{71} from this set. The left part of Figure C.2 shows how this set of twelve takes can be excused in parallel. The scheduling problem of these twelve tasks can be viewed as the scheduling problem on two independent parallel three-processor systems (i.e., without processor 4), similar

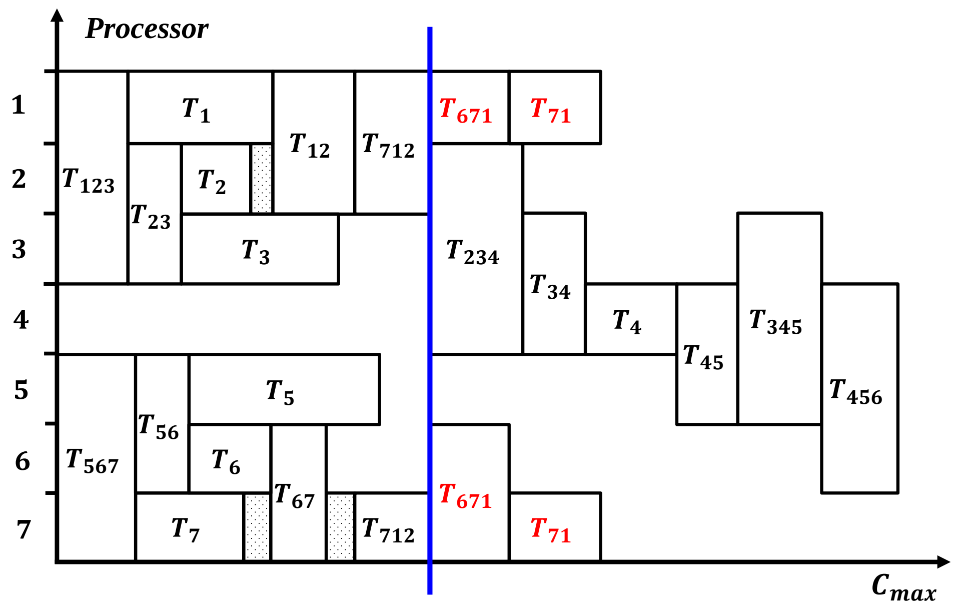


Figure C.2: A seven-node bidirectional ring with shortest path routing where $\mathcal{S}_1 \geq \mathcal{S}_2$

to the one depicted in Figure 4.1(b). This scheduling problem corresponds to the SA problem on the clockwise direction of the seven-node after removing the link corresponding to processor 4 and the six traffic demands using that link.

Based on Lemma 4.2.1, we notice these twelve tasks can be scheduled optimally. Let $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_6$, and $\mathcal{C}_7$ denote the completion time corresponding to the processors 1, 2, 6, and 7, respectively in the left part of Figure C.2. Define $\mathcal{S}_1 = max\{\mathcal{C}_1, \mathcal{C}_2\}$ and $\mathcal{S}_2 = max\{\mathcal{C}_6, \mathcal{C}_7\}$. We now show that depending on the relationship between $\mathcal{S}_1$ and $\mathcal{S}_2$, we can always construct a schedule of ratio 2.

- $\mathcal{S}_1 \geq \mathcal{S}_2$
  We append T_{712} to the end of the schedule in the left part of Figure C.2 whereby the structure of the optimum solution will not be violated. Let $OPT' = \mathcal{S}_1 + T_{712}$ be the makespan of this schedule; then, $OPT' \leq OPT$. Now, consider the six remained tasks that require processor 4. These tasks can be scheduled back-to-back without any gaps, as shown in the right part of Figure C.2. The makespan of this schedule is equal to LB.

  Once we add these tasks in the second part of the schedule, we then execute tasks T_{671} and T_{71} in parallel with them as shown in Figure C.2. Tasks T_{671} and T_{71} would essentially fit in the designed space; otherwise, the processor 4 would not be the dominant processor. Thus, the makespan of the two-part of 21 tasks schedule depicted in Figure C.2 is equal to: $OPT' + LB \leq 2 \times OPT$.

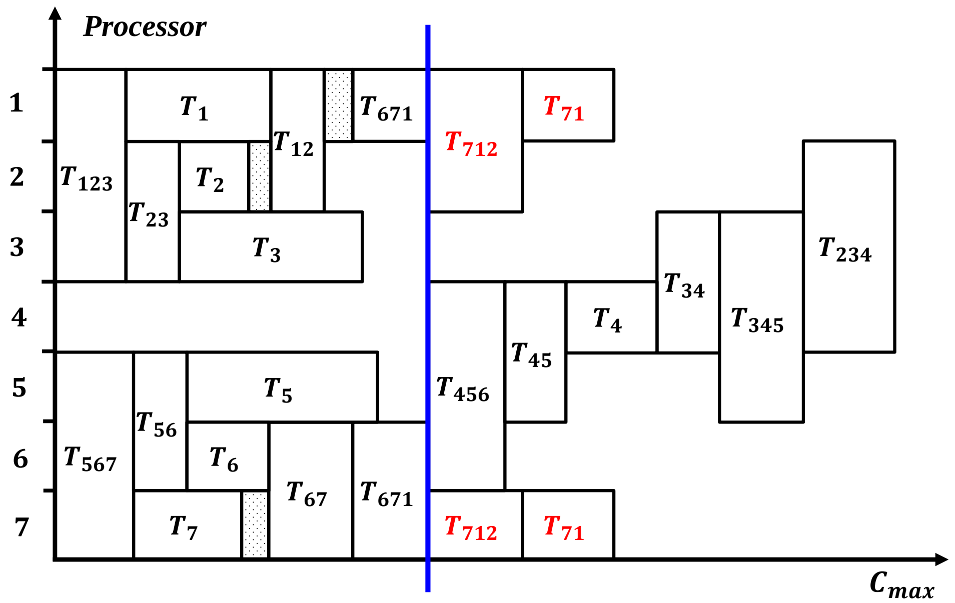


Figure C.3: A seven-node bidirectional ring with shortest path routing where $\mathcal{S}_1 < \mathcal{S}_2$

- $\mathcal{S}_1 < \mathcal{S}_2$
  Since $\mathcal{S}_2$ is greater than $\mathcal{S}_1$, the existing schedule remains optimum after adding task T_{671} to the end of schedule in the left part of Figure C.3. Let $OPT' = \mathcal{S}_2 + T_{671}$ be the makespan of this schedule; then, $OPT' \leq OPT$.

  Similarly, if we schedule the six remained tasks that require processor 4 back-to-back without any gaps as shown in the right part of Figure C.3, we observe T_{712} and T_{71} can be also executed in parallel with them. Concretely, we know that $T_{712} \leq LB - T_{234}$; otherwise, processor 1, 2, or 7 would be the dominant processor. As the makespan of the second part of the schedule is equal to LB, the total completion time of the two-part of 21 tasks schedule shown in Figure C.3 is equal to: $OPT' + LB \leq 2 \times OPT$.