Spectrum Assignment in Mesh Elastic Optical Networks

Mahmoud Fayez[‡], Iyad Katib[†], George N. Rouskas[⋆†], H. M. Faheem[•‡]
[†]King Abdulaziz University, [⋆]North Carolina State University, [‡]Fujitsu, [•]Ain Shams University

***Abstract*—Spectrum assignment has emerged as the key design and control problem in elastic optical networks. We have shown that spectrum assignment in networks of general topology is a special case of scheduling multiprocessor tasks on dedicated processors. Based on this insight, we develop and evaluate efficient and effective algorithms for mesh and chain networks that build upon list scheduling concepts.**

I. Introduction

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].

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), a modulation format that has been widely adopted in broadband wireless and copper-based communication systems, and 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 refer to [9].

OFDM, a multiple-carrier modulation scheme, 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 a key challenge and the routing and spectrum assignment (RSA) problem has emerged as an essential network design and control problem [13], [14].

In offline RSA, the input typically consists of a set of forecast 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). Several variants of the RSA problem have been studied in the literature that take into account various design aspects including the reach versus modulation level (spectral efficiency) tradeoff [15], traffic grooming [16], and restoration [17]. These problem variants are NP-hard, as RSA is a generalization of the well-known routing and wavelength assignment (RWA) problem [18]. Therefore, while most studies provide integer linear program (ILP) formulations for the RSA variant they address, they propose heuristic algorithms for solving medium to large problem instances. Such *ad hoc* solution approaches have two drawbacks. First, they do not provide insight into the structure of the optimal solution and hence cannot be easily adapted to other problem variants. Second, it is quite difficult to characterize the performance of heuristic algorithms, and our recent work has demonstrated that heuristics for the related RWA problem produce solutions that are far away from optimal even for problem instances of moderate size [19]. For a survey of spectrum management techniques in elastic optical

networks, including a review of solution approaches to RSA problem variants, we refer the reader to our recent survey [20].

In this paper, we build upon well-understood scheduling theory techniques to develop efficient and effective algorithms for spectrum assignment in mesh optical networks. In Section II, we review our earlier work that provides insight into the properties and structure of the spectrum assignment problem as a special case of a general multiprocessor scheduling problem, in which a task must be executed by multiple machines simultaneously. In Section III, we develop algorithms for multiprocessor scheduling that can be used to solve the corresponding spectrum assignment problem; we present two algorithms, one for general topology (mesh) networks, and a faster one for chain networks. We present numerical results to evaluate the performance of the algorithms in Section IV, and we conclude the paper in Section V.

II. SA in Mesh Networks: A Special Case of Multiprocessor Scheduling

We consider the following general definition of the spectrum assignment (SA) problem in elastic optical networks.

- *SA Inputs:* (1) a graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{A}$ is the set of arcs (directed edges); (2) a spectrum demand matrix $T = [t_{sd}]$, such that t_{sd} is the number of spectrum slots required to carry the traffic from source s to destination d; and (3) a fixed route r_{sd} from node s to node d.
- *SA Objective:* for each traffic demand, assign spectrum slots along all the arcs of its route such that the total required amount of spectrum used on any arc in the network is minimized.
- *RSA Constraints:* (1) spectrum contiguity: each demand is assigned contiguous spectrum slots; (2) spectrum continuity: each demand uses the same spectrum slots along all arcs of its route; and (3) non-overlapping spectrum: demands that share an arc are assigned non-overlapping parts of the available spectrum.

Now, consider the multiprocessor scheduling problem $P|fix_j|C_{max}$, defined as [21]:

- $P|fix_j|C_{max}$ *Inputs:* a set of m identical processors, a set of n tasks, the processing time p_j of task j, and a set fix_j of processor sets that will execute each task j.
- $P|fix_j|C_{max}$ *Objective:* schedule the tasks so as to minimize the makespan $C_{max} = \max_j C_j$ of the schedule, where C_j indicates the completion time of task j.
- $P|fix_j|C_{max}$ *Constraints:* (1) no preemption is allowed; (2) all the processors in the selected set must work on task j simultaneously, and (3) each processor may execute at most one task at any given time.

In earlier work [22], we have proved that the SA problem in mesh networks transforms to the $P|fix_j|C_{max}$ multiprocessor scheduling problem, but the reverse is not true. In other words, SA is a special case of $P|fix_j|C_{max}$, and hence, any algorithm for the latter problem may also solve the former. A formal proof of the transformation is omitted due to page

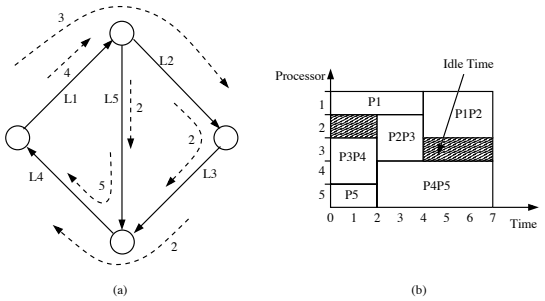

Fig. 1. (a) Instance of the SA problem on a mesh network with five directed links (arcs). (b) Optimal schedule of the corresponding $P|fix_j|C_{max}$ problem

constraints. In the transformation, each arc in the SA problem maps to a processor in the scheduling problem, each traffic demand to a task, the number of spectrum slots of a demand to the processing time 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 SA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|fix_j|C_{max}$. Furthermore, the spectrum contiguity constraint of SA is equivalent to the no preemption constraint of $P|fix_j|C_{max}$, 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.

As an example, Figure 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 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 $L5$.

It has been shown [21] 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 [23].

By building upon this new perspective, it was shown in [22] that (1) $P3|fix_j|C_{max}$ transforms to the SA problem in a unidirectional ring with three links, hence the latter is NP-

hard, and (2) the SA problem is solvable in polynomial time on chain networks with at most three links, but is NP-hard on chains with four or more links. The latter result confirms the conclusion in [24] that the SA problem is harder than the wavelength assignment problem which can be solved in polynomial time on chains of any length. In [22], we also developed a suite of list scheduling algorithms specifically designed for the SA problem in chains; the algorithms are both fast and effective, in that they produce solutions that, on average, are within 5% of the lower bound.

III. Scheduling Algorithms for Spectrum Assignment in Mesh Networks

We now present a new efficient scheduling algorithm for the $P|fix_j|C_{max}$ problem, which may also be used to solve the SA problem in mesh networks. The input to the algorithm is a list of n tasks, $j = 1, \ldots, n$, along with their corresponding processing times, p_j, and sets of processors, fix_j. Tasks in the list may be sorted in different order; in this work, we consider and compare two distinct orders:

- *Longest-First (LF):* tasks appear in the list in decreasing order of their processing time p_j.
- *Widest-First (WF):* tasks are listed in decreasing order of the size $|fix_j|$ of their processor set.

In either case, we assume that ties are broken arbitrarily. We also define two processor sets as *compatible* if they are disjoint, in which case the two processor sets can be scheduled simultaneously.

Figure 2 presents a pseudocode description of the scheduling algorithm for the $P|fix_j|C_{max}|$ problem. Depending on whether the tasks are listed in longest-first or widest-first order, we will refer to the scheduling algorithm as *SA-LF* or *SA-WF*, respectively[1].

The algorithm maintains an array of m booleans to keep track of the set of free processors, initialized to all processors, and a list of in-progress tasks, initialized to the empty set. Initially, the tasks in the imput list L are sorted in longest-first or widest-first order. The algorithm then repeatedly calls two procedures, *ScheduleTasks()* and *AdvanceTime()*, until all the tasks in list L have been scheduled (i.e., until L becomes empty).

The *ScheduleTasks()* procedure takes as arguments the list L of unscheduled tasks, the list L_p of in-progress tasks, and the set F of free processors at time t. It then considers tasks in L one at a time in an attempt to schedule them starting at time t; note that a task j can be scheduled if all processors in fix_j are free at time t, i.e., it is pairwise compatible with all in-progress tasks. Every task that can be scheduled is marked as running in list L_p of in-progress tasks. This process continues until either the end of list L is reached or all processors become busy. At that point, procedure *AdvanceTime()* is called. This procedure finds the first in-progress task that ends, and

[1]In this notation, we use the acronym "SA" to emphasize both that this is a *scheduling algorithm* for the $P|fix_j|C_{max}$ problem, and the fact that this algorithm solves the *spectrum assignment* problem.

Scheduling Algorithm (SA-LF/WF) for $P|fix_j|C_{max}$
Input: A list L of n tasks on m processors, each task j having a processing time p_j and a set $fix_j \subseteq \{1, 2, \ldots, m\}$ 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. Sort the tasks in list L based on longest-first or widest-first criteria
2. $L_p[1, \ldots, n] \leftarrow false$ //The list of in-progress tasks
3. $t \leftarrow 0$ //Scheduling instant
4. $F_p \leftarrow m$ //Counter of free processors
5. $Counter \leftarrow 0$ //Counter of finished tasks
6. $F[1, \ldots, m] \leftarrow true$ //The list of idle(free) processors
7. **while** $Counter \neq n$ **do**
8. $\quad j \leftarrow 0$
9. $\quad$ ScheduleTasks(L, L_p, F, t, j)
10. $\quad$ AdvanceTime($L_p, F, t, Counter$)
11. **end while**
12. **return** the task start times S_j

end

Procedure ScheduleTasks(L, L_p, F, t, j)
Operation: Schedules as many tasks from the input list L to start execution at time t, and moves these tasks from L to the list of in-progress tasks L_p

begin
1. **while** $j \neq n$ **and** $F_p > 0$ **do**
2. $\quad$ **if** $L_{p_j} = false$ **and** $F_{fix_j} = true$ **then**
3. $\quad\quad S_j \leftarrow t$ // Task j starts execution at time t
4. $\quad\quad L_{p_j} = true$
5. $\quad\quad F_{fix_j} = false$
6. $\quad\quad F_p \leftarrow F_p - count(fix_j)$
7. $\quad\quad$ ScheduleTasks($L, L_p, F, t, j + 1$)
8. $\quad\quad$ **break**
9. $\quad$ **endif**
10. $\quad j \leftarrow j + 1$
11. **end while** // no more tasks may start at time t

end

Procedure AdvanceTime($L_p, F, t, Counter$)
Operation: Finds the first task or tasks to complete after time t, removes them from the list of in-progress tasks, and advances time to the time these tasks end

begin
1. $j \leftarrow 0$
2. $j_{min} \leftarrow$ -1 //Index of earliest task to finish
3. $t_{min} \leftarrow \infty$ //the default value to find minimum finish time
4. **while** $j \neq n$ **do**
5. $\quad$ **if** $L_{p_j} = true$ **and** $S_j + p_j > t$ **and** $S_j + p_j < t_{min}$ **then**
6. $\quad\quad j_{min} \leftarrow j$
7. $\quad\quad t_{min} \leftarrow S_j + p_j$
8. $\quad$ **endif**
8. $\quad j \leftarrow j + 1$
9. **end while**
10. $F_{fix_{j_{min}}} \leftarrow true$ //Set processors to free again
11. $F_p \leftarrow F_p + count(fix_{j_{min}})$
12. $t \leftarrow t_{min}$ //Advance time

end

Fig. 2. The scheduling algorithm SA-LF/WF for $P|fix_j|C_{max}$ and the corresponding spectrum assignment problem

advances the time to the time t' this task ends. It then frees all invovled processors for this task. Consequently, the procedure *ScheduleTasks()* is called again to schedule any remaining tasks starting at time t', and this process repeats until all tasks have been scheduled.

Each of the two procedures *ScheduleTasks()* and *AdvanceTime()* is called n times in the worst case, where n is the number of tasks. In the worst case, the *ScheduleTasks()* will consider all n tasks in list L, and for each task it will check whether its processor set is a subset of the set F of free processors (line 2), and if so, it will remove these processors from the set (line 5); these operations take time $O(m)$ in the worst case, where m is the number of processors. Since all other operations are constant, the running time of this procedure is $O(nm)$. Procedure *AdvanceTime()* checks all in-progress tasks to identify the ones with minimum finish time, and frees their processors; therefore, its worst-case running time is $O(m+n)$. Therefore, the overall running time complexity of the algorithm is $O(mn^2)$.

A. Scheduling Algorithm for Chain Networks

In the special case of chain networks, the corresponding scheduling problem is such that the m processors, each corresponding to a link of the chain, can be labeled linearly as $1, \ldots, m$. Furthermore, the processors required by each task are contiguous and may be represented as a range, a fact that has two implications. First, checking whether a task's processors are free at some time t can be performed in constant time, rather than in time $O(m)$ as in the general case of mesh networks. Second, assigning processors to a task naturally divides the previous free range of processors into (at most) two parts: one part consisting of free processors with labels smaller than that of the processor with the lowest label required by the task, and one consisting of free processors with labels larger than that of the processor with the highest label required by the task. Therefore, it is possible to schedule tasks by recursively searching the (at most) two free ranges of processors created when a task is scheduled.

The recursive version of *ScheduleTasks()* for chain networks is shown in Figure 3. Since, as we mentioned above, the operations in steps 2 and 5 now take constant time, the worst-case running time of the procedure for chain networks is $O(n)$. Therefore, the overall running time of the scheduling algorithm is $O(n^2)$, and is independent of the number m of processors (or links of the chain).

Procedure ScheduleTasks(L, L_p, F, t)
Operation: Schedules as many tasks from the input list L to start execution at time t, and moves these tasks from L to the list of in-progress tasks L_p

```
begin
begin
  1.  while j ≠ n and F_p > 0 do
  2.      if L_{p_j} = false and F_{fix_j} = true then
  3.          S_j ← t // Task j starts execution at time t
  4.          L_{p_j} = true
  5.          F_{fix_j} = false
  6.          F_p ← F_p − count(fix_j)
  8.          F_l ← new left range of free processors
  9.          F_r ← new right range of free processors
  7.          ScheduleTasks(L, L_p, F_l, t, j + 1)
  7.          ScheduleTasks(L, L_p, F_r, t, j + 1)
  8.          break
  9.      endif
  10.     j ← j + 1
  11. end while  // no more tasks may start at time t
end
```

Fig. 3. A specialized version of the *ScheduleTasks()* procedure for the $P|fix_j|C_{max}$ problem corresponding to a chain network

IV. Numerical Results

In this section, we present the results of simulation experiments we have carried out to evaluate the performance of the scheduling algorithms for mesh and chain networks. We assume that the elastic optical 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).

In our experiments, we also used various other probability values for both the skewed low and skewed high distributions, but the results are very similar to the ones shown below.

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 [25, Table 1]. We then transform the SA problem instance to the equivalent instance of $P|fix_j|C_{max}$, and run the scheduling algorithm described in the previous section to schedule the tasks.

In order to evaluate the performance of our algorithms, and since the optimal solution is not known due to the fact that $P|fix_j|C_{max}$ is NP-complete, we compute the lower bound as follows. 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

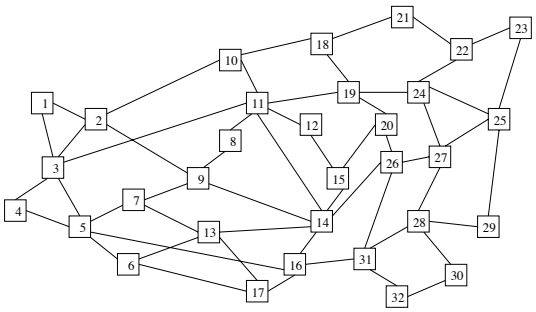

Fig. 4. The 32-node, 108-link topology used in the experiments

Π_k denote the sum of processing times of tasks that require processor k:

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

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

$$LB = \max_{k=1,\ldots,m} \{\Pi_k\}. \quad (2)$$

We then compute the ratio of the makespan produced by the algorithm to this 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.

A. Mesh Networks

We have carried out simulation experiments on three network topologies[2]: (1) a 10-node, 32-link network; (2) the 32-node, 108-link network shown in Figure 4; and (3) a 75-node, 200-link topology. For each network topology, we generate random traffic demands between each source-destination pair[3] using each of the three distributions described earlier. Each demand is routed over the shortest path between its source and destination nodes.

The results for the 10-node network are shown in Figure 5. All problem instances resulted in the optimal solution except, In four problem instances, the SA-LF algorithm found solutions with a makespan about 10% higher than the lower-bound, while for the remaining instances the solutions constructed by the algorithm had a makespan equal to the lower bound and hence, they are optimal. For the 32- and 75-node networks, the SA-LF produced optimal solutions for all three traffic distributions and all random problem instances we generated.

Figures 6, and 7 show the performance of the SA-LF and SA-WF algorithms, respectively, on complete mesh networks of varying sizes, in which traffic between each pair of nodes is routed over a randomly chosen path. Each point in the figures is the average of 30 randomly generated problem instances for

[2]The number of links of each network topology refers to directional links (arcs) since each direction of a link is considered independently for the purpose of spectrum assignment.

[3]The number of traffic demands for the 10-node, 32-node, and 75-node networks are 90, 992, and 5550, respectively.

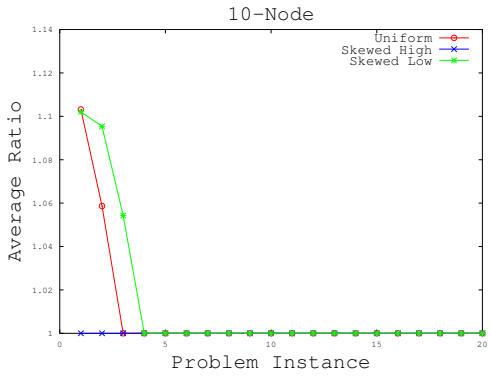

Fig. 5. Average ratio of makespan to lower bound for 10-node network

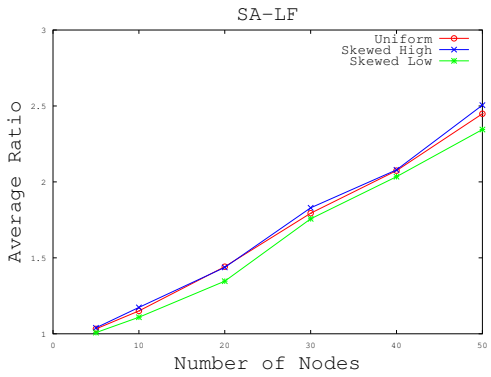

Fig. 6. Average ratio of makespan to lower bound, complete mesh, SA-LF

the given network size and distribution. The performance of the two algorithms is similar and is not affected significantly by the traffic distribution (uniform, skewed high, or skewed low). The reason that the ratio of makespan to lower bound increases with the size of the network is due to the fact that, with a Note that, for a complete mesh, the optimal solution would be for each demand to take the single-link shortest path to its destination. However, since each demand is routed along a randomly selected path, and the lengths of these random paths increase with the size of the network, such a solution will move further from the lower bound (optimal) as the network size increases. The performance of the algorithms in the two figures simply reflects this observation.

B. Chain Networks

Figures 8 and 9 show results for chain networks of varying sizes and the SA-LF and SA-WF algorithms, respectively. We observe that the SA-LF algorithm performs better for all three traffic distributions, and produces results that are within 5% of the lower bound. The SA-WF algorithm, which considers tasks for scheduling based on the number of processors they require, may pair long tasks with short ones, thus creating gaps that result in a longer makespan.

V. Concluding Remarks

We have developed scheduling algorithms that solve efficiently the spectrum assignment problem in mesh networks

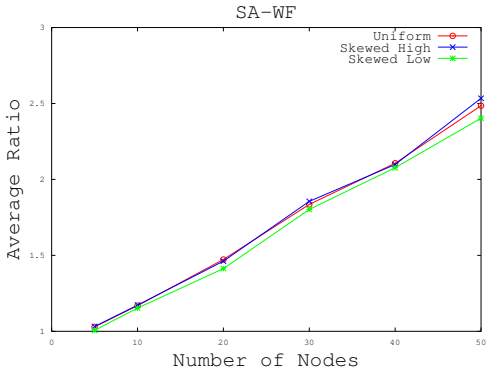

Fig. 7. Average ratio of makespan to lower bound, complete mesh, SA-WF

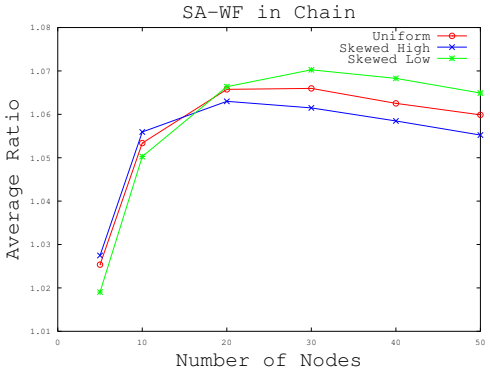

Fig. 9. Average ratio of makespan to lower bound, chain netw. with SA-WF

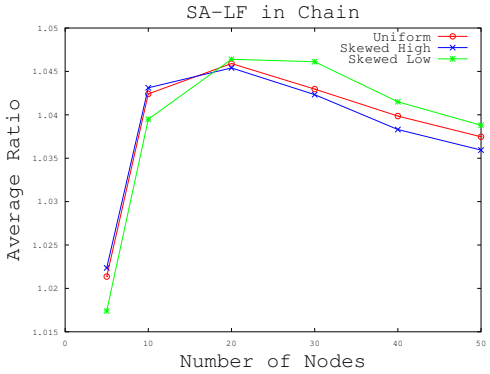

Fig. 8. Average ratio of makespan to lower bound, chain netw. with SA-LF

with good performance, under the assumption that traffic demands are routed over a fixed path. Our current research efforts are directed towards algorithms that tackle jointly the routing and spectrum assignment problems.

VI. Acknowledgments

This work was supported in part by the National Science Foundation under Grant CNS-1113191, and in part by the High Performance Computing Project in King Abdulaziz University.

References

[1] P. J. Winzer, "Beyond 100G Ethernet," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 26–30, July 2010.

[2] A. N. et. al, "Optical network design with mixed line rates," *Optical Switching and Networking*, vol. 6, no. 3, pp. 227–237, 2009.

[3] I.-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*, vol. 50, no. 5, pp. 82–89, 2012.

[5] M. J. et. al, "Elastic and adaptive optical networks: possible adoption scenarios and future standardization aspects," *IEEE Communications Magazine*, vol. 49, no. 10, pp. 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 ECOC*, September 2009, p. 7.7.4.

[7] O. G. et. al, "Elastic optical networking: a new dawn for the optical layer?" *IEEE Comm. Magazine*, vol. 50, no. 2, pp. s12–s20, 2012.

[8] M. J. et. al, "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 ECOC*, September 2008, p. Th.3.F.6.

[9] G. Z. et. al, "A survey on OFDM-based elastic core optical networking," *IEEE CST*, vol. 15, no. 1, pp. 65–87, First Quarter 2013.

[10] W. Shieh, "OFDM for flexible high-speed optical networks," *Journal of Lightwave Technology*, vol. 29, no. 10, pp. 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*, vol. 14, no. 6, pp. 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*, vol. 25, no. 1, pp. 131–138, January 2007.

[13] M. Klinkowski and K. Walkowiak, "Routing and spectrum assignment in spectrum sliced elastic optical path network," *IEEE Communications Letters*, vol. 15, no. 8, pp. 884–886, 2011.

[14] 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*, 2011, pp. 1503–1511.

[15] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos, "Elastic bandwidth allocation in flexible OFDM–based optical networks," *Journal of Lightwave Technology*, vol. 29, no. 9, pp. 1354–1366, 2011.

[16] 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)*, March 2011, p. OTuI1.

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

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

[19] Z. Liu and G. N. Rouskas, "A fast path-based ILP formulation for offline RWA in mesh optical networks," in *Proceedings of IEEE GLOBECOM 2012*, December 2012, pp. 2990–2995.

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

[21] J. A. Hoogeveen, S. L. V. de Velde, and B. Veltman, "Complexity of scheduling multiprocessor tasks with prespecified processor allocations," *Discree Applied Mathematics*, vol. 55, pp. 259–272, 1994.

[22] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas, "Spectrum assignment in optical networks: A multiprocessor scheduling perspective," *Journal of Optical Communications and Networking*, vol. 6, no. 8, pp. 754–763, August 2014.

[23] 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.

[24] S. Shirazipourazad, C. Zhou, Z. Derakhshandeh, and A. Sen, "On routing and spectrum allocation in spectrum-sliced optical networks," in *Proceedings of IEEE INFOCOM*, April 2013, pp. 385–389.

[25] M. J. et. al, "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network," *IEEE Communications Magazine*, vol. 48, no. 8, pp. 138–145, 2010.