

Offline Distance-Adaptive Routing and Spectrum Assignment (DA-RSA) in Rings

Sahar Talebi[†], Iyad Katib^{*}, George N. Rouskas^{†*}

[†]Operations Research and Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 USA

^{*}King Abdulaziz University, Jeddah, Saudi Arabia

Abstract—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. We first show that the distance-adaptive routing and spectrum assignment (DA-RSA) problem in mesh networks is a special case of a multiprocessor scheduling problem. We then develop a suite of efficient and effective DA-RSA algorithms that build upon list scheduling concepts. Our work explores the tradeoffs involved in DA-RSA algorithm design, and opens up new research directions that may leverage the vast literature in scheduling theory.

I. INTRODUCTION

Optical networking has a vital role in the operation of the global Internet and the availability of reliable and survivable communication services. Conventional fixed-grid WDM technology assigns a full wavelength to each traffic demand, even small ones, resulting in low utilization of the available spectrum [1]. This issue is even more challenging when transmitting higher data rates over long distance [2]. Elastic optical networks [2], [3] have been introduced in response to the need to accommodate the ever growing traffic demands within a finite spectrum capacity. Using finer spectrum granularity, elastic networks enable flexibility in allocating spectrum resources proportionally to the traffic demand size.

Routing and spectrum assignment (RSA) arises as the fundamental design and control problem in elastic networks. Several aspects of the problem have been studied in the literature, including offline RSA [4], [5], online RSA [6], [7], distance adaptive RSA (DA-RSA) [8], [9], fragmentation-aware RSA (FA-RSA) [10], RSA and traffic grooming [11], and RSA with restoration [12]; for a recent survey of the literature, we refer the reader to [13].

Although operators are in the process of transitioning their networks to mesh topologies, large portions of the current infrastructure are built on ring topologies. Hence, RSA algorithms for rings will be important in the short- and medium-term; importantly, such solutions are likely to provide insight into extending the techniques to mesh networks. Therefore, there has been increasing interest in RSA solutions for ring networks within the research community. The study in [14] considered the case of dynamic traffic flows between every pair of nodes, and showed that employing elastic spectrum

allocation in ring topologies increases spectrum utilization and reduces the blocking ratio compared with fixed-grid WDM technology. A distance adaptive elastic optical ring network with traffic grooming was considered in [15]. The joint routing, spectrum assignment, and modulation format problem were formulated as an integer linear problem (ILP), and was solved with heuristic algorithms. This study also provided upper bounds for both the spectral minimization and transceiver minimization problems.

Several studies have addressed theoretical aspects of the RSA problem in ring topologies. For instance, an algorithm with a $(4 + 2\epsilon)$ -approximation ratio for ring networks was presented in [16]. In [17], it was shown that the contiguity (i.e., adjacency) constraint in the spectrum assignment problem is redundant, in that it can be constructed from the optimal solution to the wavelength assignment (i.e., corresponding coloring) problem. A comprehensive study on the complexity and approximation ratios for the spectrum assignment (SA) and routing and spectrum assignment (RSA) problems in rings is available in our recent work [18].

In this paper, we leverage scheduling theory to provide new insight into the structure of the offline distance-adaptive RSA (DA-RSA) problem and to present efficient and effective algorithms for rings. In Section II, we introduce the general result that DA-RSA with fixed alternate routing in general mesh networks is a special case of a 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 III. In Section IV, we present the results of experiments to compare the performance of the algorithms with respect to the lower bound, and we conclude the paper in Section V.

II. OFFLINE DA-RSA IN MESH NETWORKS: A SPECIAL CASE OF MULTIPROCESSOR SCHEDULING

Distance-adaptive (DA) spectrum allocation, a concept first introduced in [19], 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 [9]. The distance-adaptive routing and spectrum assignment (DA-RSA) problem

This work was supported by the National Science Foundation under Grant CNS-1113191, and in part by the Deanship of Scientific Research (DSR), King Abdulaziz University, under Grant No. 2-611-1434-HiCi.

with fixed-alternate routing in mesh elastic optical networks can be defined as:

Definition 2.1 (DA-RSA): Given a directed graph $G = (V, E)$ with V vertices (nodes) and E arcs (directed edges), k alternate routes, $r_{sd}^1, \dots, r_{sd}^k$, from each node s to each node d , and traffic demand matrix $T = [t_{sd,l}]$ in which $t_{sd,l}$ represents the required amount of spectrum to transmit traffic from source node s to destination node d along the l -th route, $l = 1, \dots, k$, select one of possible route for each traffic demand and assign required spectrum on all the edges of this route such that the total amount of spectrum in the network is minimized while the following three constraints are satisfied:

- *spectrum contiguity constraint:* each demand is assigned contiguous spectrum on all the edges of each route.
- *spectrum continuity constraint:* each demand is assigned the same spectrum along all the edges of its route.
- *non-overlapping spectrum constraint:* demands that share an edge 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 RSA problem reduces to the spectrum assignment (SA) problem. In recent work [20], we have proved that the 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 following definition of $P|fix_j|C_{max}$ is adapted from [21], [22].

Definition 2.2 ($P|fix_j|C_{max}$): Given a set of m identical processors, a set of n tasks with processing time $p_j, j = 1, \dots, n$, and a prespecified set fix_j of processors for executing each task $j, j = 1, \dots, n$, schedule these tasks under three constraints: (1) preemption is not allowed; (2) each task must be executed by all of its set of required processors fix_j at the same time; and (3) a processor can process at most one task at a time, so as to minimize the makespan of the tasks denoted by $C_{max} = \max_j C_j$ where C_j stands for the completion time of task j .

If the number of processors m is fixed and given in advance, then the problem is denoted by $Pm|fix_j|C_{max}$. The proof that SA transforms to $P|fix_j|C_{max}$ is available in [20].

Consider now the more general multiprocessor scheduling problem $P|set_j|C_{max}$, defined as follows [23], [24]:

Definition 2.3 ($P|set_j|C_{max}$): Given a set of m identical processors, a set of n tasks, a prespecified set $set_j = \{fix_j^1, \dots, fix_j^k\}$ of k alternative processor sets to execute each task j , and processing time p_j^l for executing task j on set fix_j^l , 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 $P|set_j|C_{max}$.

Lemma 2.1: DA-RSA with fixed-alternate routing in mesh networks transforms to $P|set_j|C_{max}$.

Proof. Consider an instance of the RSA problem with fixed-alternate routing on a directed topology graph $G = (\mathcal{V}, \mathcal{E})$, a set of k routes $\{r_{sd}^1, \dots, r_{sd}^k\}$ for each source-destination pair (s, d) , and demand matrix $T = [t_{sd,l}], l = 1, \dots, k$. 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 $a_i \in \mathcal{E}$, (2) there is a task j for each source-destination pair (s, d) , (3) there is a $set_j = \{fix_j^1, \dots, fix_j^k\}$ for each task j with $fix_j^l = \{q : a_q \in \{r_{sd}^l\}\}$ where (s, d) is the source-destination pair corresponding to task j , and (4) the processing time of task j on processor set fix_j^l is $p_j^l = t_{sd,l}, l = 1, \dots, k$. In other words, each alternate path transforms to the corresponding alternate set of processors, while the amount of spectrum on that path define the corresponding processing time in the scheduling problem.

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 RSA problem is equivalent to the completion time of the last task executed on the corresponding processor. Accordingly, minimizing the spectrum use on any arc of the RSA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|set_j|C_{max}$. ■

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

Clearly, the $P|fix_j|C_{max}$ problem is a special case of $P|set_j|C_{max}$ 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 $P|set_j|C_{max}$ 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.

In the context of the $P|fix_j|C_{max}$ problem, we refer to tasks as *compatible* if they can be executed simultaneously, i.e., they do not share any processors. More formally, we have the following definition.

Definition 2.4 (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., $fix_i \cap fix_j = \emptyset, \forall i, j \in \mathcal{T}$.

A. 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 $O(m^2)$.

III. OFFLINE 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 $P|set_j|C_{max}$ 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 $P = NP$ [25]. The two-processor problem $P2|set_j|C_{max}$ has been proved in [26] to be NP-hard. 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. There are two broad approaches to solve this problem [13]. 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 three 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 three algorithms utilize the concept of compatible tasks to minimize the makespan, C_{max} , of the corresponding scheduling problem.

A. 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 $Pm|fix_j|C_{max}$ 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

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 $set_j = \{fix_j^1, \dots, fix_j^k\}$ of k alternative processor sets with its corresponding processing time $p_j = \{p_j^1, \dots, p_j^k\}$ and $A_l = [a_1, \dots, a_m]$ 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 $fix_j \subseteq \{1, 2, \dots, m\}$ of required processors

begin

```

1.  $F \leftarrow [0, \dots, 0]_{1 \times m}$  //Completion time of each of  $m$  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.      $alt_z \leftarrow F + p_j^z A_z$ 
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.        $F \leftarrow alt_w$ 
15.        $fix_j \leftarrow fix_j^w$ 
16.        $p_j \leftarrow p_j^w$ 
17. end while
end

```

Fig. 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

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 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 problem 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 determined by either the SP or TLB algorithms, the original $Pm|set_j|C_{max}$ problem has been reduced to the $Pm|fix_j|C_{max}$ problem. In [20], 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 in [20], 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; for the details on the operation of LFC, the reader is referred to [20]. 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 $O(n^2)$.

B. A Joint Routing and Spectrum Assignment Algorithm

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 an R+SA algorithm may be longer than necessary.

In this section, we propose a new algorithm that makes routing decisions jointly with spectrum assignment. The algorithm is a variant of the well-known class of list scheduling algorithms in that it takes as input a list of tasks, processes the list sequentially, and builds the schedule one task at a time, as it encounters the tasks in the list. However, our algorithm differs 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 this algorithm as *set scheduling* (SS). Second, the list is not built once at the beginning of the algorithm; rather, it is built incrementally during the execution of the algorithm, as we explain shortly.

The 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, $L = [(1, fix_1^1), (1, fix_1^2), (2, fix_2^1), (2, fix_2^2), \dots]$ 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.

Due to page constraints, we have not included a pseudocode description of the SS algorithm. Its running time complexity is $O(n^3)$, where n is the number of tasks.

IV. NUMERICAL RESULTS

We now describe the experiments we have carried out to compare the performance of the three DA-RSA algorithms in bidirectional ring networks with $N = 5, 7, 9, 11, 13, 15$ 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) [2], [19]. Thus, we assume that the slot width is 12.5 GHz, and there are two modulation formats as represented in [19]:

- 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), while 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. Note that, since the $P2N|set_j|C_{max}$ problem corresponding to a bidirectional ring network with N nodes is NP-hard for $N \geq 4$ [18], 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 2-4 plot the average ratio of the three algorithms, denoted by SP+LFC, TLB+LFC, and SS, 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 1.2, i.e., it is always within 20% 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, SP+LFC outperforms TLB+LFC in most cases regardless of the demand distribution, with the exception of small ring networks. Note that in the small ring networks of up to seven nodes in which TLB+LFC is better than SP+LFC, and based on the modulation formats we consider, a specific demand requires the same number of slots regardless of whether it is routed on the shortest or non-shortest path. This behavior indicates that, in ring networks in which there are only two possible paths, whenever demands routed along the non-shortest path require a larger number of slots than along the shortest path, it is preferable to use shortest path routing as an R+SA solution. This observation is especially true in large rings where SP+LFC is clearly the best solution, since in large networks selecting the non-shortest path incurs a spectrum penalty along a large number of links. Finally, we observe that the performance of the SS algorithm relative to the two R+SA algorithms depends strongly on the demand distribution. In particular, for distance-independent traffic and small- to medium-size rings, the SS algorithm is able to find solutions using non-shortest paths that outperform both R+SA algorithms. At the other extreme, a strategy that uses non-shortest path does not work well under the distance-increasing traffic distribution. Overall, our results indicate that (1) due to the spectrum penalty of long paths, a strategy that uses shortest path routing is a clear winner for large rings, and (2) for small- and medium-size rings the performance of the algorithms depends on the traffic distribution.

V. CONCLUDING REMARKS

We have shown that the DA-RSA problem transforms to a processor scheduling problem, and we have developed list scheduling algorithms for ring networks. 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. Overall, the best

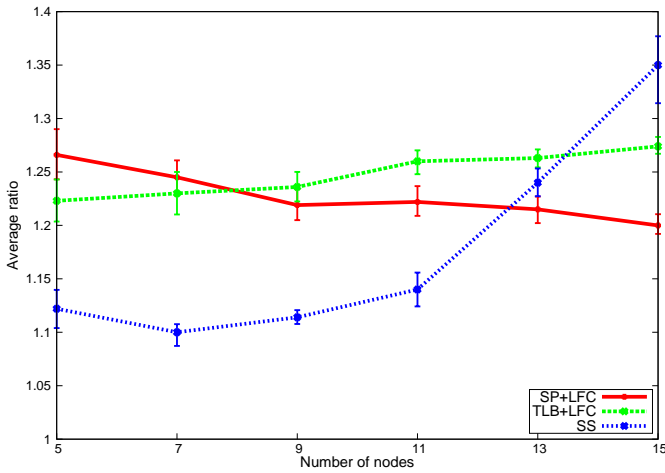


Fig. 2. Average ratio vs. number of ring nodes, distance-independent distribution

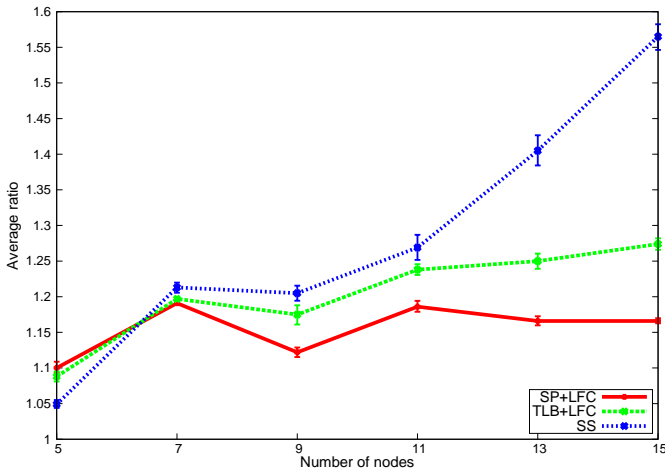


Fig. 3. Average ratio vs. number of ring nodes, distance-increasing distribution

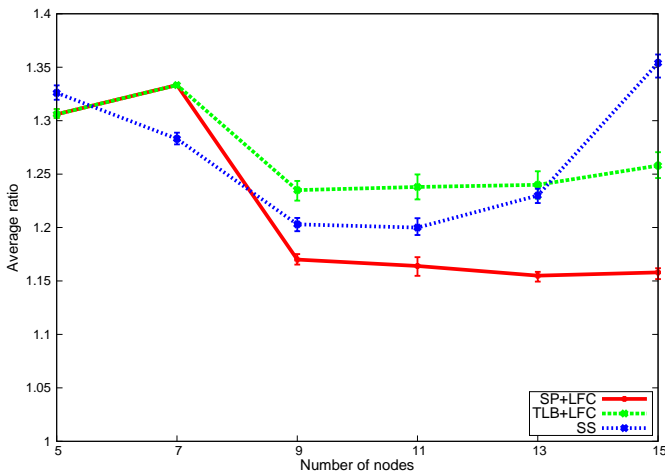


Fig. 4. Average ratio vs. number of ring nodes, distance-decreasing distribution

algorithm is always within 10-20% of the lower bound, indicating that scheduling concepts can be successfully adapted to address network design problems. Our current research focuses on extending these techniques to mesh networks.

REFERENCES

- [1] 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.
- [2] O. Gerstel *et al.* Elastic optical networking: a new dawn for the optical layer? *IEEE Communications Magazine*, 50(2):s12–s20, 2012.
- [3] M. Jinno *et al.* Dynamic optical mesh networks: Drivers, challenges and solutions for the future. *ECOC*, Paper 7.7.4, September 2009.
- [4] Y. Wang, X. Cao, and Y. Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. *IEEE INFOCOM*, pages 1503–1511, 2011.
- [5] A. N. Patel *et al.* Routing, wavelength assignment, and spectrum allocation algorithms in transparent flexible optical WDM networks. *Optical Switching and Networking*, 9(3):191–204, 2012.
- [6] X. Wan *et al.* Dynamic routing and spectrum assignment in flexible optical path networks. *OFC/NFOEC*, Paper JWA55, March 2011.
- [7] X. Wang *et al.* Blocking performance in dynamic flexible grid optical networks — what is the ideal spectrum granularity? *ECOC*, Paper Mo.2.K.6, September 2011.
- [8] B. Kozicki *et al.* Distance-adaptive spectrum allocation in elastic optical path network (SLICE) with bit per symbol adjustment. *OFC/NFOEC*, Paper OMU3, March 2010.
- [9] S. Yang and F. Kuipers. Impairment-aware routing in translucent spectrum-sliced elastic optical path networks. *NOC*, June 2012.
- [10] K. Wen *et al.* Dynamic on-demand lightpath provisioning using spectral defragmentation in flexible bandwidth networks. *ECOC*, Paper Mo.2.K.4, September 2011.
- [11] Y. Zhang *et al.* Traffic grooming in spectrum-elastic optical path networks. *OFC/NFOEC*, Paper OTu11, March 2011.
- [12] A. Eira *et al.* Optimized design of shared restoration in flexible-grid transparent optical networks. In *OFC/NFOEC*, Paper JTh2A37, 2012.
- [13] 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*, 13:34–48, July 2014.
- [14] F. Musumeci *et al.* Dynamic grooming and spectrum allocation in optical metro ring networks with flexible grid. *ICTON*, Paper We.A1.2, June 2013.
- [15] C. Rottondi *et al.* Routing, modulation level, and spectrum assignment in optical metro ring networks using elastic transceivers. *Journal of Optical Communications and Networking*, 5(4):305–315, 2013.
- [16] S. Shirazipourzad *et al.* On routing and spectrum allocation in spectrum-sliced optical networks. *INFOCOM*, pp. 385–389, April 2013.
- [17] I. Popescu *et al.* On the optimal design of a spectrum-switched optical network with multiple modulation formats and rates. *Journal of Optical Communications and Networking*, 5(11):1275–1284, 2013.
- [18] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas. On routing and spectrum assignment in rings. *Journal of Lightwave Technology*, 33(1):151–160, January 2015.
- [19] M. Jinno *et al.* Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network. *IEEE Communications Magazine*, 48(8):138–145, 2010.
- [20] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas. Spectrum assignment in optical networks: A multiprocessor scheduling perspective. *J. Optical Commun. and Netw.*, 6(8):754–763, Aug. 2014.
- [21] E. Bampis *et al.* Scheduling of independent dedicated multiprocessor tasks. *ISAAC 2002*, volume LNCS 2518, pages 391–402, 2002.
- [22] J. Hoogeveen *et al.* Complexity of scheduling multiprocessor tasks with prespecified processor allocations. *Discr. Appl. Math.*, 55:259–272, 1994.
- [23] J. Chen and Ch. Lee. General multiprocessor task scheduling. *Naval Research Logistics*, 46(1):57–74, 1999.
- [24] K. Jansen and L. Porkolab. General multiprocessor task scheduling: Approximate solutions in linear time. *SIAM Journal on Computing*, 35(3):519–530, 2005.
- [25] L. Torres *et al.* On the approximability of multiprocessor task scheduling problems. *ISAAC 2002*, volume LNCS 2518, pages 403–415, 2002.
- [26] M. Kubal. The complexity of scheduling independent two-processor tasks on dedicated processors. *Info. Proc. Letters*, 24(3):141–147, 1987.