IET Networks

Research Article

IET Journals
The Institution of Engineering and Technology

Distance-adaptive routing and spectrum assignment in rings

ISSN 2047-4954
Received on 11th September 2015
Revised on 17th November 2015
Accepted on 15th December 2015
doi: 10.1049/iet-net.2015.0085
www.ietdl.org

Sahar Talebi[1], Iyad Katib[2], George N. Rouskas[1,2] ✉

[1]*Operations Research and Department of Computer Science, North Carolina State University, Raleigh, USA*
[2]*Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia*
✉ *E-mail: rouskas@ncsu.edu*

Abstract: Distance adaptive spectrum allocation exploits the tradeoff between spectrum width and reach to improve resource utilisation by tailoring the modulation format to the level of impairments along the path. The authors first show that the distance-adaptive routing and spectrum assignment (DA-RSA) problem is a special case of a multiprocessor scheduling problem. The authors then develop a suite of efficient and effective DA-RSA algorithms for ring networks, that build upon list scheduling concepts. This 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.

1 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 utilisation 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 [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 utilisation 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 (SA), and modulation format problem were formulated as an in integer linear problem, and was solved with heuristic algorithms. This study also provided upper bounds for both the spectral minimisation and transceiver minimisation problems.

Several studies have addressed theoretical aspects of the RSA problem in ring topologies. For instance, an algorithm with a $(4+2\varepsilon)$-approximation ratio for ring networks was presented in [16]. In [17], it was shown that the contiguity (i.e. adjacency) constraint in the SA problem is redundant, in that it can be constructed from the optimal solution to the wavelength assignment (i.e. corresponding colouring) problem. A comprehensive study on the complexity and approximation ratios for the SA and 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 DA-RSA problem and to present efficient and effective algorithms for rings. In Section 2, 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 3. In Section 4, we present the results of experiments to compare the performance of the algorithms with respect to the lower bound (LB), and we conclude the paper in Section 5.

2 DA-RSA as 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 utilisation 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 DA-RSA problem 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, $r_{sd}^1, \ldots, r_{sd}^k$, from each node s to each node d, and traffic demand matrix $\boldsymbol{T}=[t_{sd,l}]$ in which $t_{sd,l}$ represents the required amount of spectrum to transmit traffic from source node s to destination node d along the lth route, $l=1, \ldots, k$, 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 minimised 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 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 $P|\text{fix}_j|C_{\max}$. That is, the SA problem can be transformed to $P|\text{fix}_j|C_{\max}$, but the reverse is not always true. Based on this reduction, any algorithm that solves the $P|\text{fix}_j|C_{\max}$ problem also solves the SA problem. The following definition of $P|\text{fix}_j|C_{\max}$ 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, $j = 1, \ldots, n$, and a prespecified set fix_j of processors for executing each task j, $j = 1, \ldots, n$, schedule theses 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 minimise 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 $Pm|\text{fix}_j|C_{\max}$. The proof that SA transforms to $P|\text{fix}_j|C_{\max}$ is available in [20].

Consider now the more general multiprocessor scheduling problem $P|\text{set}_j|C_{\max}$, 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 $\text{set}_j = \{\text{fix}_j^1, \ldots, \text{fix}_j^k\}$ 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: (i) preemption is not allowed; (ii) each task j is processed by exactly one set of processors in set_j simultaneously; and (iii) each processor can execute at most one task at each time, so as to minimise 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 $P|\text{set}_j|C_{\max}$.

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

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 $\{r_{\text{sd}}^1, \ldots, r_{\text{sd}}^k\}$ for each source-destination pair (s, d), and demand matrix $\boldsymbol{T} = [t_{\text{sd},l}]$, $l = 1, \ldots, k$. It is possible to build an instance of $P|\text{set}_j|C_{\max}$ such that: (i) there is a processor i for every arc in $a_i \in \mathcal{E}$, (ii) there is a task j for each source-destination pair (s, d), (iii) there is a $\text{set}_j = \{\text{fix}_j^1, \ldots, \text{fix}_j^k\}$ for each task j with $\text{fix}_j^l = \{q: a_q \in \{r_{\text{sd}}^l\}\}$ where (s, d) is the source-destination pair corresponding to task j, and (iv) and processing time of task j on processor set fix_j^l is $p_j^l = t_{\text{sd},l}$, $l = 1, \ldots, k$. 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 an 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, minimising the spectrum use on any arc of the RSA problem is equivalent to minimising the makespan of the schedule in the corresponding problem $P|\text{set}_j|C_{\max}$.

We also note that it can be shown by counter-example that the reverse of the above lemma is not true, that is, $P|\text{Set}_j|C_{\max}$ does not transform to DA-RSA and hence, it is a more general problem.

Clearly, the $P|\text{fix}_j|C_{\max}$ problem is a special case of $P|\text{set}_j|C_{\max}$ 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 $P|\text{set}_j|C_{\max}$ problem reduces to $P|\text{fix}_j|C_{\max}$, in which case any algorithm that solves the latter problem may be applied to schedule the tasks.

In the context of the $P|\text{fix}_j|C_{\max}$ problem, we refer to tasks as *compatible* if they can be executed simultaneously, that is, 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 $P|\text{fix}_j|C_{\max}$ problem are said to be compatible if and only if their prespecified sets of processors are pairwise disjoint, that is, $\text{fix}_i \cap \text{fix}_j = \emptyset, \forall i, j \in \mathcal{T}$.

2.1 Lower bound for ring networks

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 LB. To this end, we note that the amount of flow across any cut of the network is a LB 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 $N!/(N-2)!2!$ two-link cuts, hence a LB 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 LB can also be expressed as $O(m^2)$.

3 DA-RSA algorithms for rings

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 $P|\text{set}_j|C_{\max}$ 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 $P|\text{set}_j|C_{\max}$ unless $P = NP$ [25]. The two-processor problem $P2|\text{set}_j|C_{\max}$ 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 RSA 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 $R + \text{SA}$ in the literature. A second approach is to make RSA decisions jointly.

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

3.1 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 $Pm|\text{fix}_j|C_{\max}$ problem. The algorithms only differ in how they make the routing decision, or, from the point of view of

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. $F \leftarrow [0, \ldots, 0]_{1\times m}$ //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 *TLB algorithm to select one set fix_j for executing each task j of the $Pm|set_j|C_{max}$ problem*

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 Fig. 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 is $k = 2$, 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 $Pm|\text{set}_j|C_{\max}$ problem has been reduced to the $Pm|\text{fix}_j|C_{\max}$ problem. In [20], we introduced a suite of list scheduling algorithms for solving the latter problem (i.e. for performing the SA) 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 $Pm|\text{fix}_j|C_{\max}$ 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 $O(n^2)$.

3.2 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 RSA 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 two new algorithms that make routing decisions jointly with SA. 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:

(i) *Task selection:* A subset of the input set of tasks is selected.
(ii) *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.
(iii) *Task scheduling:* The list is scanned and tasks are considered for inclusion in the schedule. Scheduled tasks are removed from further consideration.
(iv) *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.

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 $set_j = \{fix_j^1, \ldots, fix_i^k\}$ of k alternative processor sets with its corresponding processing time $p_j = \{p_j^1, \ldots, p_j^k\}$
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. $F \leftarrow \{1, \ldots, m\}$ //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. $fix_j \leftarrow fix_j^z$
13. $p_j \leftarrow p_j^z$
14. $F \leftarrow F \setminus fix_j$
15. $S_j \leftarrow t$ // Task j starts execution at time t
16. Remove task j from $\mathcal{S}$
17. **break**
18. **if** $fix_j = \emptyset$ **then** Add task j to list R
19. **end while**
20. $\mathcal{T} \leftarrow$ a subset of $\mathcal{S}$ determined by the source/sink cut algorithm
21. $L \leftarrow$ a sorted list of $\mathcal{T}$
22. **if** $L = R$ **then** // Performs optimization for all the remaining tasks
23. $L \leftarrow$ a sorted list of $\mathcal{S}$
24. **while** list $L \neq \emptyset$ or $F \neq \emptyset$ **do**
25. $l \leftarrow$ first task in list L; Remove task l from list L
26. **for** $z = 1$ to k
27. **if** $fix_l^z \subseteq F$ **then**
28. $fix_l \leftarrow fix_l^z$
29. $p_l \leftarrow p_l^z$
30. $F \leftarrow F \setminus fix_l$
31. $S_l \leftarrow t$ // Task l starts execution at time t
32. Remove task j from $\mathcal{S}$
33. **break**
34. **end while**
35. $i \leftarrow$ the first task executing at time t to complete
36. $t \leftarrow S_i + p_i$
37. $F \leftarrow F \cup fix_i$
38. **end while**
39. **return** the task start times S_j

end

10

Fig. 2 *SS algorithm for* $Pm|set_j|C_{max}$

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 LB 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 LB, it is important to minimise 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, that is, $|\text{fix}_j^1| \leq |\text{fix}_j^2|$, 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, $L = [(1, \text{fix}_1^1), (1, \text{fix}_1^2), (2, \text{fix}_2^1), (2, \text{fix}_2^2), \ldots]$ 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

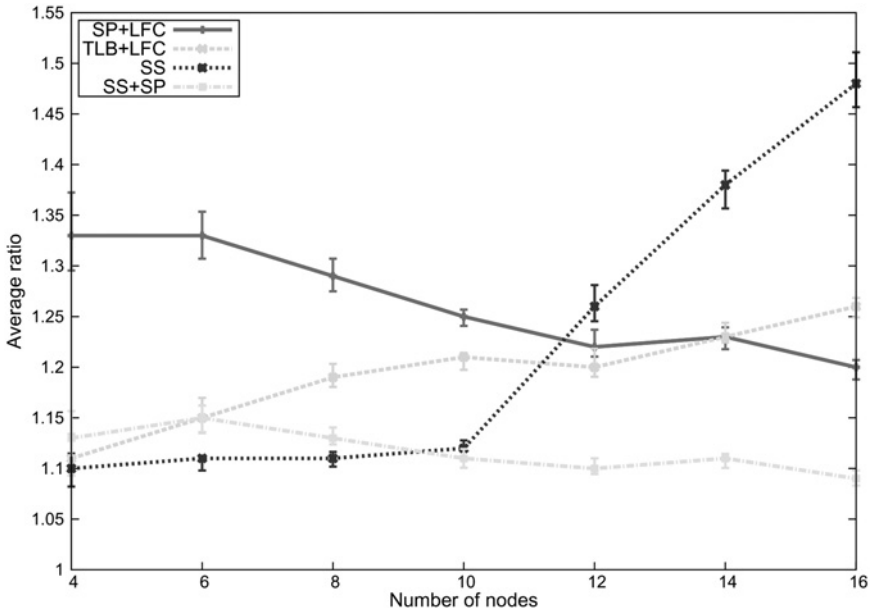

Fig. 3 *Average ratio against number of nodes, distance-independent distribution*

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 DA 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 $(1, \text{fix}_1^1)$ is the first item in list L). We then remove from the list both task-processor set pairs $(1, \text{fix}_1^1)$, $(1, \text{fix}_1^2)$, 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 SA 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.

3.2.1 Pseudocode description of the SS algorithm: A pseudocode description of the SS algorithm is presented in Fig. 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

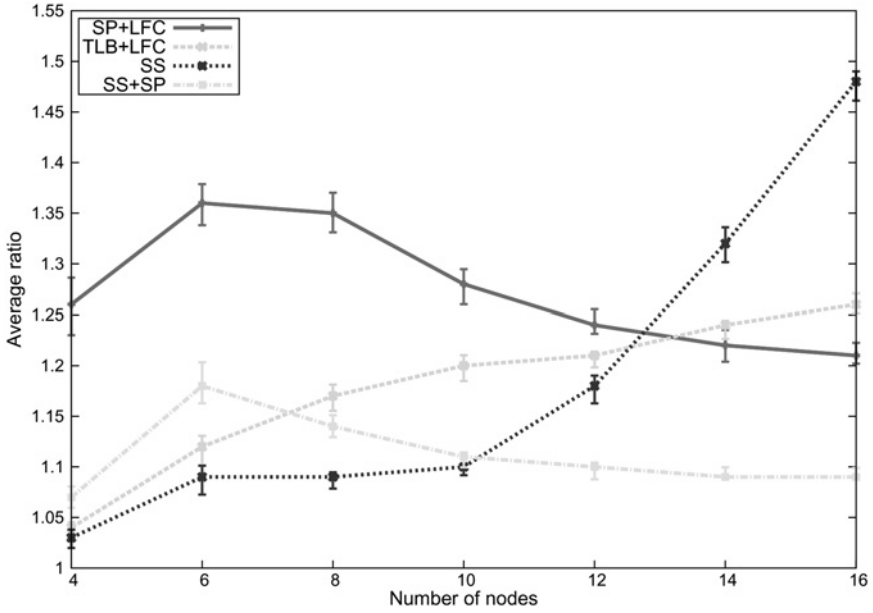

Fig. 4 *Average ratio against number of nodes, distance-increasing distribution*

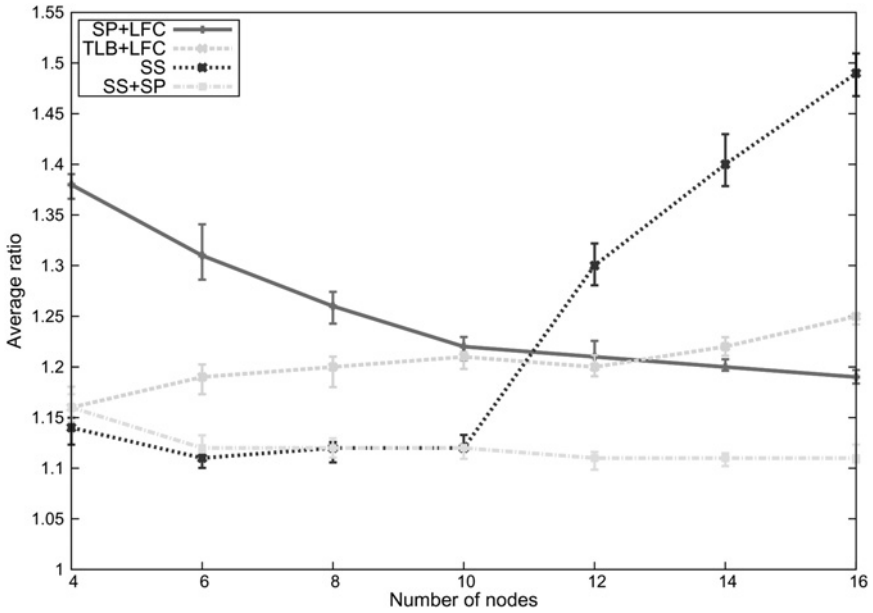

Fig. 5 *Average ratio against number of nodes, distance-decreasing distribution*

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.

3.2.2 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 smallest processor sets are skipped, and scheduled later, that is, either during the optimisation phase of the algorithm (which starts in Line 22 of the pseudocode shown in Fig. 2) or after the time t is updated.

4 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 $N = 4, 6, 8, 10, 12, 14, 16$ 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) [2, 19]. As in [27], we assume that the ring diameter does not exceed 100 Km, a reasonable value for rings covering metropolitan areas. Further, following [15], we make the assumption that all links are of equal length, hence the reach of each modulation format can be expressed in number of links. Thus, we assume that the slot width is 12.5 GHz, and the two modulation formats represented in [19] are sufficient for the metro ring sizes:

- 16-QAM modulation format for paths with up to eight 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 eight 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 LB (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 $P2N|\text{set}_j|C_{\max}$ 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 LB; therefore, the performance of the algorithms with respect to the optimal may be better than this ratio indicates. Nevertheless, this metric accurately characterises the relative performance of the algorithms.

Figs. 3–5 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, that is, it is always within 15% of the LB on the amount of spectrum required to route all demands. Since the (unknown) optimal solution will generally lie above the LB, these results indicate that our algorithms are effective in constructing solutions close to optimal one.

Another important observation is that of the two $R+\mathrm{SA}$ 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 $R+\mathrm{SA}$ algorithms, but does not work well for larger ring networks. On the 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 (i) due to the spectrum penalty of long paths, strategies that give preference to shortest path routing work best for large rings, and (ii) 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.

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

6 Acknowledgment

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

7 References

1 Shen, G., Zukerman, M.: 'Spectrum-efficient and agile CO-OFDM optical transport networks: architecture, design, and operation', *IEEE Commun. Mag.*, 2012, **50**, (5), pp. 82–89
2 Gerstel, O., Jinno, M., Lord, A., *et al.*: 'Elastic optical networking: a new dawn for the optical layer?', *IEEE Commun. Mag.*, 2012, **50**, (2), pp. s12–s20
3 Jinno, M., Takara, H., Kozicki, B.: 'Dynamic optical mesh networks: drivers, challenges and solutions for the future'. Proc. 35th European Conf. on Optical Communication (ECOC), September 2009, p. 7.7.4
4 Wang, Y., Cao, X., Pan, Y.: 'A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks'. Proc. of IEEE INFOCOM, 2011, pp. 1503–1511
5 Patel, A.N., Ji, Ph.N., Jue, J.P., *et al.*: 'Routing, wavelength assignment, and spectrum allocation algorithms in transparent flexible optical WDM networks', *Opt. Switch. Netw.*, 2012, **9**, (3), pp. 191–204
6 Wan, X., Wang, L., Hua, N., *et al.*: 'Dynamic routing and spectrum assignment in flexible optical path networks'. Proc. of Optical Fiber Communication Conf. and the National Fiber Optic Engineers Conf. (OFC/NFOEC), March 2011, p. JWA55
7 Wang, X., Zhang, Q., Kim, I., *et al.*: 'Blocking performance in dynamic flexible grid optical networks – what is the ideal spectrum granularity?'. Proc. 37th European Conf. on Optical Communication (ECOC), September 2011, p. Mo.2.K.6
8 Kozicki, B., Takara, H., Sone, Y., *et al.*: 'Distance-adaptive spectrum allocation in elastic optical path network (SLICE) with bit per symbol adjustment'. Proc. of Optical Fiber Communication and National Fiber Optic Engineers Conf. (OFC/NFOEC), March 2010, p. OMU3
9 Yang, S., Kuipers, F.: 'Impairment-aware routing in translucent spectrum-sliced elastic optical path networks'. Proc. 17th European Conf. on Networks and Optical Communications (NOC), June 2012
10 Wen, K., Yin, Y., Geisler, D.J., *et al.*: 'Dynamic on-demand lightpath provisioning using spectral defragmentation in flexible bandwidth networks'. Proc. 37th European Conf. on Optical Communication (ECOC), September 2011, p. Mo.2.K.4
11 Zhang, Y., Zheng, X., Li, Q., *et al.*: 'Traffic grooming in spectrum-elastic optical path networks'. Proc. of Optical Fiber Communication Conf. and the National Fiber Optic Engineers Conf. (OFC/NFOEC), March 2011, p. OTuI1
12 Eira, A., Pedro, J., Pires, J.: 'Optimized design of shared restoration in flexible-grid transparent optical networks'. Proc. of OFC/NFOEC 2012, 2012, p. JTh2A37
13 Talebi, S., Alam, F., Katib, I., *et al.*: 'Spectrum management techniques for elastic optical networks: a survey', *Opt. Switch. Netw.*, 2014, **13**, pp. 34–48
14 Musumeci, F., Puleio, F.F., Tornatore, M.: 'Dynamic grooming and spectrum allocation in optical metro ring networks with flexible grid'. Proc. of ICTON 2013, June 2013, p. We.A1.2
15 Rottondi, C., Tornatore, M., Pattavina, A., *et al.*: 'Routing, modulation level, and spectrum assignment in optical metro ring networks using elastic transceivers', *IEEE/OSA J. Opt. Commun. Netw.*, 2013, **5**, (4), pp. 305–315
16 Shirazipourazad, S., Zhou, Ch., Derakhshandeh, Z., *et al.*: 'On routing and spectrum allocation in spectrum-sliced optical networks'. Proc. of IEEE INFOCOM, April 2013, pp. 385–389
17 Popescu, I., Cerutti, I., Sambo, N., *et al.*: 'On the optimal design of a spectrum-switched optical network with multiple modulation formats and rates', *IEEE/OSA J. Opt. Commun. Netw.*, 2013, **5**, (11), pp. 1275–1284
18 Talebi, S., Bampis, E., Lucarelli, G., *et al.*: 'On routing and spectrum assignment in rings', *J. Lightw. Technol.*, 2015, **33**, (1), pp. 151–160
19 Jinno, M., Kozicki, B., Takara, H., *et al.*: 'Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network', *IEEE Commun. Mag.*, 2010, **48**, (8), pp. 138–145
20 Talebi, S., Bampis, E., Lucarelli, G., *et al.*: 'Spectrum assignment in optical networks: a multiprocessor scheduling perspective', *J. Opt. Commun. Netw.*, 2014, **6**, (8), pp. 754–763
21 Bampis, E., Caramia, M., Fiala, J., *et al.*: 'Scheduling of independent dedicated multiprocessor tasks'. Proc. 13th Annual Int. Symp. on Algorithms and Computation, 2002, volume LNCS 2518, pp. 391–402
22 Hoogeveen, J.A., Van de Velde, S.L., Veltman, B.: 'Complexity of scheduling multiprocessor tasks with prespecified processor allocations', *Discree Appl. Math.*, 1994, **55**, pp. 259–272
23 Chen, J., Lee, Ch.: 'General multiprocessor task scheduling', *Nav. Res. Logist.*, 1999, **46**, (1), pp. 57–74
24 Jansen, K., Porkolab, L.: 'General multiprocessor task scheduling: approximate solutions in linear time', *SIAM J. Comput.*, 2005, **35**, (3), pp. 519–530
25 Torres, L., Miranda, A., Chen, J.: 'On the approximability of multiprocessor task scheduling problems'. Proc. 13th Annual Int. Symp. on Algorithms and Computation, 2002, volume LNCS 2518, pp. 403–415
26 Kubal, M.: 'The complexity of scheduling independent two-processor tasks on dedicated processors', *Inf. Process. Lett.*, 1987, **24**, (3), pp. 141–147
27 Yang, H.-S., Herzog, M., Maier, M., *et al.*: 'Metro WDM networks: performance comparison of slotted ring and AWG star networks', *IEEE J. Sel. Areas Commun.*, 2004, **8**, pp. 1107–1116