A Scalable Solution to Network Design Problems: Decomposition with Exhaustive Routing Search

Mahmoud Fayez[•], Iyad Katib[†], George N. Rouskas[⋆†], Tarek F. Gharib[•], H. K. Ahmed[•], H.M. Faheem[•]
[†]King Abdulaziz University, [⋆]North Carolina State University, [•]Ain Shams University

***Abstract*—Many network design problems encompass two tasks, routing and resource allocation, that are so intricately intertwined as to contribute significantly to the intractability of such problems. In this paper, we make two contributions to addressing general network design problems of this nature. First, we present a new decomposition method that optimally decouples resource allocation from routing, making it possible to tackle each of these aspects separately. Second, we develop a recursive branch-and-bound algorithm to search the routing space exhaustively, yet in a scalable manner. We apply our method to a well-known intractable problem in optical networks, routing and spectrum assignment (RSA). Our results indicate that the recursive algorithm is able to search efficiently the entire routing space of topologies representative of large-scale wide area networks.**

I. Introduction

Network design problems are complex problems that arise naturally in the planning, engineering and deployment of the Internet infrastructure. Consequently, the overall operation and economic model of the Internet, and its capacity to deliver reliable and critical communication services, rely crucially on efficient and effective solutions to a range of network design problems. Similar observations apply to optical networks that constitute the foundation of the backbone (long-haul) and regional (metro-area) parts of the global infrastructure, and are now reaching into the access part in the form of PON architectures [1].

Over the past three decades, researchers have investigated an abundance of optical network design problems, including routing and wavelength assignment (RWA) [2]–[5], traffic grooming [6], [7], routing and spectrum assignment (RSA) [8], [9], and network survivability [10]. Whereas integer linear program (ILP) formulations may in theory yield exact solutions to these problems, in practice ILP models do not scale to instances typical of commercial networks. Consequently, the literature is replete with heuristics that were developed with large networks in mind; unfortunately, in general there is little information about the accuracy of solutions produced by heuristic algorithms [11]. Moreover, heuristic designs are usually one-off, carefully crafted to solve a specific problem rather than apply to a more general class of problems. Even if adapting such a heuristic to variants of the original problem were feasible, applying the modifications would require expertise in both the domain of the problem (e.g., network design, graph theory) and a range of related disciplines that include mathematical programming, operations research, and discrete optimization. Therefore, most approaches currently impose significant demands on computational resources and human expertise, and severely limit our ability to investigate the sensitivity of design decisions to problem inputs, including forecast traffic demands or capital and operating cost considerations.

This work was supported by the National Science Foundation under Grant CNS-1907142 and by the High Performance Computing Project at King Abdulaziz University.

Our work is motivated by the observation that many network design problems encompass two tasks, routing and resource allocation, that are so intricately intertwined as to contribute significantly to the intractability of such problems. Therefore, we make two contributions to addressing general network design problems of this nature. First, we present a new decomposition method that optimally decouples resource allocation from routing, making it possible to tackle each of these aspects separately. Second, we develop a recursive branch-and-bound algorithm to search the routing space exhaustively, yet in a scalable manner. Although we apply our method to routing and spectrum assignment (RSA), a well-known intractable problem in optical networks, our approach has broad applications as it decouples *optimally* the routing and resource allocation components of the optimization.

The remainder of the paper is organized as follows. In Section II, we present a formulation of a generic network design problem that we use to illustrate our approach, and we introduce a new decomposition algorithm for this problem in Section III. We develop a scalable recursive method for searching the routing space in Section IV, and we present experimental results for the RSA problem in Section V. We conclude the paper in Section VI.

II. A Reference ILP Formulation

We consider network optimization problems that may be expressed using ILP formulations and are NP-hard [5]. There are two general types of ILP models, link-based [12] and path-based [13]. In the former, the entities of interest (that is, decision variables) relate to network links; in the latter, variables relate to end-to-end paths. With a link-based model, the optimization problem is expressed as a multicommodity flow formulation, and the solver is forced to consider the entire space of network paths between any two nodes. On the other hand, path-based formulations take as input a (usually small) set of paths between each pair of nodes. Hence, path-based ILP models produce a solution that is optimal only within the input set of paths; such solution is no better than the optimal solution obtained by a link-based formulation of the same problem.

In this paper, we consider the following reference ILP formulation of a generic *routing and resource allocation* (RRA) optimization problem $\mathcal{P}_{RRA}$. Although this is a path-based formulation, we discuss shortly how it may be used to produce optimal solutions equivalent to a link-based formulation.

The reference optimization problem $\mathcal{P}_{RRA}$

- **Input:**
  - *Network topology:* a connected graph $G = (V, A)$ where V denotes the set of nodes and A denotes the set of arcs (directed links) in the network.
  - *Traffic demands:* a traffic demand matrix $T = [t_{sd}]$, where t_{sd} is a non-negative value representing the amount of traffic from node s to node d, in some appropriate units (e.g., bandwidth, wavelengths, spectrum slots, etc).
  - *Routing paths:* a set K_{sd} of paths for source-destination pair (s, d), such that all traffic from s to d is constrained to take path(s) in this set only; the paths are generated in advance and their number may vary based on the source-destination pair.
- **Constraints:**
  - A set of constraints on available network resources (e.g., link and/or switch capacity) and/or on the routing of traffic demands.
- **Metric:** A performance metric of interest, e.g., the amount of network resources to be deployed.
- **Output:**
  - *Routing:* a routing of demands over network paths.
  - *Resource allocation:* an assignment of network resources (e.g., wavelength or block of spectrum slots) to each demand.
- **Objective:** The performance metric of interest is optimized while all constraints are satisfied.

Note that, while the above is a path-based formulation, if we let K_{sd} be the set of *all possible* paths (rather than a subset of all paths) between nodes s and d, then the solver will be forced to consider all paths in the network just as with a link-based formulation. Therefore, we will use the above reference formulation as representing both classes of formulations, path- or link-based. Of course, to make this equivalent to a link-based formulation, one would have to generate all paths between every pair of nodes in the network; this number is exponential in the size of the network [14]. We address this challenge later in this work.

In general, optimization problems represented by $\mathcal{P}_{RRA}$ are NP-hard and exact solutions may be obtained only for networks of small size [11]; for Internet-scale topologies, it is not possible to solve the ILP model optimally in a reasonable amount of time. Consequently, network design problems of this nature are typically solved using heuristic algorithms; for instance, surveys of heuristic approaches for a collection of RSA problem variants are available in [8], [9]. Furthermore, column generation and/or decomposition techniques may be used to manipulate the ILP model so as to tackle a principal challenge, i.e., the large numbers of network paths for each traffic demand that need to be considered in an optimal solution.

Decomposition algorithms reduce the problem by dividing it into two simpler ILP models that are then solved in sequence [15]. The first model addresses only the routing aspect of the original problem, and it is designed so as to produce paths that are likely to lead to a good overall solution. The second ILP model is concerned only with resource allocation along the paths calculated by the first model. The two ILP models are solved iteratively until no further improvement to the performance metric of interest can be achieved. Whereas such an iterative process often yields good solutions, in general the decomposition into two sequential subproblems is not optimal, and in certain circumstances it may fail to even produce feasible solutions.

Column generation is another iterative technique that aims to obtain an optimal solution without considering all network paths [16], [17]. Rather, a new path is generated at the end of an iteration only if it is determined that it would lead to an improvement of the solution currently at hand. A recent study [18] presented a new column generation model that uses lightpath configurations and can achieve a significant speedup. Another study [19] developed a branch-and-price algorithm and combined it with additional techniques, including relaxations, cuts, and heuristics. These studies [18], [19] are able to solve larger RSA problem instances than commercial solvers by carefully crafting solutions to the specific problem and objective considered. By contrast, the decomposition algorithm we present next is both intuitive and generic and hence, applicable to a wide range of network design problems and objectives.

III. Decomposition Algorithm for Problem $\mathcal{P}_{RRA}$

Consider now the problem $\mathcal{P}_{RA}$, derived from the reference problem $\mathcal{P}_{RRA}$ we defined in the previous section by providing as input only a single path for each pair of nodes in the network, i.e., $|K_{sd}| = 1$ for $\mathcal{P}_{RA}$. In other words, the routing of demands is fixed and not part of the optimization process, hence $\mathcal{P}_{RA}$ is purely a resource allocation (RA) problem. Intuitively, the solution space of problem $\mathcal{P}_{RA}$ is significantly smaller than that of problem $\mathcal{P}_{RRA}$, and hence we expect the former to be "easier" to solve than the latter[1]. This observation motivates us to explore a solution to $\mathcal{P}_{RRA}$ that decomposes the problem into routing and resource allocation sub-problems that are tackled independently but differently than previous decomposition approaches.

Let us define a *routing configuration* $R_i, i = 1, \ldots, L$, as an assignment of one path to each pair of nodes in the network, whereby the path assigned to pair (s, d) is selected among the set K_{sd} of paths input to problem $\mathcal{P}_{RRA}$. Let L be the number of distinct routing configurations for $\mathcal{P}_{RRA}$, and note that

$$L = \Pi_{s,d}|K_{sd}| \quad (1)$$

may be a very large integer. Consider now the following decomposition algorithm for problem $\mathcal{P}_{RRA}$:

[1]Note that we do not claim that problem $\mathcal{P}_{RA}$ has lower asymptotic complexity than problem $\mathcal{P}_{RRA}$, only that the size of the solution space that a solver must explore is considerably smaller.

Decomposition Algorithm (RRA-DA) for $\mathcal{P}_{RRA}$

1) **Preprocessing/Path Computation.** Generate a set K_{sd} of paths between each source-destination pair (s, d).
2) **Exhaustive Demand Routing.** Generate all L routing configurations $R_i, i = 1, \ldots, L$, from the path sets K_{sd}, where L is given by (1).
3) **Resource Allocation.** Solve the L resource allocation sub-problems $\mathcal{P}_{RA}(R_i), i = 1, \ldots, L$. Each problem $\mathcal{P}_{RA}(R_i)$ is derived from problem $\mathcal{P}_{RRA}$ by routing the traffic demand between a pair of nodes over the corresponding path in routing configuration R_i.
4) **Postprocessing.** Let $\mathcal{P}_{RA}(R_k)$ be the problem[2] whose solution yields the optimal objective value. Return routing configuration R_k and the resource allocation determined by this solution as the overall solution to $\mathcal{P}_{RRA}$.

Note that, unlike earlier heuristics (e.g., [15]) that decompose the problem into routing and spectrum allocation subproblems that are solved independently and sequentially, the above RRA-DA algorithm performs an *exhaustive search* over the entire routing space and returns the best among the L routing configurations. In other words, RRA-DA is optimal with respect to the set of paths generated by the preprocessing Step 1. Therefore, if all possible paths between every pair of nodes are generated at Step 1, RRA-DA is an optimal algorithm for problem $\mathcal{P}_{RRA}$. Therefore, with this exhaustive routing search, our approach represents an *exact decoupling* of the routing and resource allocation aspects of the optimization, and hence, it has broad applications.

In practice, of course, each of the three Steps 1-3 of RRA-DA poses significant computational challenges. As we mentioned earlier, the number of paths between each node pair is exponential in the size of the network; furthermore, often the resource allocation subproblem (e.g., spectrum assignment [20]) is NP-hard for general topologies. Nevertheless, it is possible to only generate a large but polynomial number of paths (e.g., using k-shortest path algorithms), whereas the resource allocation subproblem may be solved effectively on fixed paths using polynomial-time heuristics that work well in practice [21]. On the other hand, even if we use polynomial-time algorithms for path generation and resource allocation, the number L of routing configurations in (1) that have to be generated in Step 2 (and, hence, the number of resource allocation subproblems that have to be solved in Step 3) remains exponential.

1) Application to the RSA Problem: The focus of this work is to implement in a scalable manner the exhaustive search over the entire space of routing configurations *without* having to enumerate (in Step 2) or evaluate (in Step 3) all possible configurations R_i. The exhaustive search algorithm, presented in the following section, may be applied to any network design problem represented by $\mathcal{P}_{RRA}$. Nevertheless, in order to illustrate our technique, we will consider the RSA problem within the class of problems $\mathcal{P}_{RRA}$. In RSA, the traffic demands represent the number of spectrum slots for each source-destination pair; the constraints include the contiguity, continuity, and non-overlapping spectrum constraints; and the objective is to minimize the number of spectrum slots used in any network link [21]. In the following, we will use $\mathcal{P}_{RSA}$ and $\mathcal{P}_{SA}$ to refer specifically to the RSA problem and its spectrum allocation (SA) subproblem, respectively.

We apply the following well-known solutions to the path computation (Step 1) and spectrum assignment (Step 3) components of the RRA-DA decomposition algorithm:

- **Path Computation.** We apply depth-first search (DFS) to generate *all* paths between each source-destination pair in the network. Since the number of paths between each pair of nodes is not known in advance, we run DFS twice; the first time to determine the number of paths for each pair and allocate memory, and the second time to build the data structure that holds the paths. Since our goal is to implement efficiently a search over the entire space of routing configurations, using the maximum number of paths (and, hence, routing configurations) is the best way to test the scalability of the approach we present in the next section.
- **Spectrum Assignment.** For a given routing configuration R, we solve the spectrum assignment problem $\mathcal{P}_{SA}(R)$ using the *longest first-fit (LFF)* policy [21]. LFF considers each traffic demand in descending order of path length (i.e., demands with longer paths are considered first), and assigns spectrum slots to each demand along its (single) path using the first-fit allocation policy [3]. The LFF heuristic is fast and performs well when routing is fixed and the objective is to minimize the maximum amount of spectrum used on any link [21].

Note that since we are using a heuristic for the spectrum assignment subproblem, the RRA-DA algorithm may not necessarily find an optimal solution to $\mathcal{P}_{RSA}$ despite the fact that it searches the entire routing space. Nevertheless, in the context of our decomposition approach the spectrum assignment algorithm is orthogonal to the algorithm used to search the routing space, in the sense that the former is used to solve each subproblem $\mathcal{P}_{SA}(R_i)$ separately from others and hence is independent[3] of the *strategy* that the latter uses in navigating the space of routing configurations R_i. Therefore, the search algorithm we present next may be combined with any algorithm for the spectrum assignment subproblem, including optimal ones, and will deliver similar benefits.

IV. Scalable Recursive Routing Search

Let us assume that the path computation step (Step 1) of the RRA-DA algorithm has generated a set of K_{sd} paths for each node pair (s, d), and that the paths of each set are labeled as $p_{sd}^1, p_{sd}^2, \cdots, p_{sd}^{|K_{sd}|}$. Let N be the number of nodes in the network and let $k_{min} = \min_{s,d}\{|K_{sd}|\}$ be the minimum number of paths generated for any node pair. Since there are

[2]If there are multiple solutions with the same optimal value, the algorithm may return one solution chosen arbitrarily, all solutions, or a subset of solutions using criteria other than the objective value. Without loss of generality, we assume that one solution of optimal value is returned.

[3]The choice of a spectrum assignment algorithm affects both the overall running time and whether the decomposition is guaranteed to reach an optimal solution or not. But as we mentioned earlier, in this work we are only concerned with developing efficient strategies for searching the routing space.

$O(N^2)$ node pairs, expression (1) implies that L is $\Omega(k_{min}^{N^2})$. Therefore, any algorithm to enumerate all possible routing configurations R must take time that is exponential in the size of the network *even if* k_{min} is a small integer independent of the number N of nodes.

Consider now a routing configuration R and let $SOL(R)$ be the solution to the spectrum assignment problem on this configuration obtained by the LFF policy[4]. A lower bound $LB(R)$ on the solution to the spectrum assignment problem for R may be obtained by ignoring any spectrum fragmentation that may result from satisfying the spectrum contiguity and continuity constraints, and simply accounting for the fact that each link ℓ of the network must use at least as many spectrum slots as to carry all the demands whose path includes this link:

$$LB(R) \quad = \quad \max_{\ell} \left\{ \sum_{s,d:\ell \in p_{sd}} t_{sd} \right\} \quad \leq \quad SOL(R) \qquad (2)$$

Let $BestSOL$ be the best solution that has been obtained while searching the routing space at the time that we consider some configuration R. If $LB(R) \geq BestSOL$, then we know that R cannot lead to a better solution since, because of (2), $SOL(R) \geq BestSOL$. We now make two observations with respect to any routing configuration R whose lower bound is worse than the current best solution.

- We do not need to invoke the LFF policy to obtain a spectrum assignment for R. At first thought this may not appear to provide any computational savings: calculating the lower bound from (2) takes time $O(MN^2)$, where M is the number of edges in the network, the same asymptotic complexity as executing the LFF policy. However, suppose that R is derived from another configuration R' by only changing the path for a single node pair, and that $LB(R')$ is known. Then $LB(R)$ may be computed in time $O(M)$ by simply updating the demands along the links of the two paths (old and new) between this node pair, representing a significant decrease in complexity, especially for larger networks.
- More importantly, let R be a *partial* routing configuration in that it specifies paths for only a subset of node pairs and has no paths for the remaining pairs. Then, adding a path for some node pair (s, d) that is not currently represented in R may only *increase* the lower bound in (2) since the links of the path will have to carry the new demand t_{sd}. Therefore, we may terminate the search for configurations derived from the partial configuration R at this point, potentially eliminating a large fraction of the routing space from further consideration.

Based on these observations, we have developed a scalable branch-and-bound recursive routing search (SRRS) procedure, shown as Algorithm 1, to search the entire space of routing configurations. The algorithm starts from an *empty* configuration R_0, i.e., one without any paths, and recursively adds one path at a time until it reaches a complete configuration.

[4]All the observations in this section are valid regardless of the specific spectrum assignment algorithm used to obtain $SOL(R)$. We refer to the LFF policy only because it is the algorithm we use in this work.

Algorithm 1 Scalable Recursive Routing Search **SRRS**(s, d, R, LB)

Input:
s: source node
d: destination node
R: routing configuration
LB: lower bound of configuration R
$BestSOL$: best solution so far (global variable)
$BestR$: best routing configuration so far (global variable)
Output:
Best routing configuration and corresponding solution

```
1: if s > N then {R is a complete configuration}
2:    SOL = solution obtained by LFF on R;
3:    if SOL < BestSOL then
4:       //Update best known solution
5:       BestSOL = SOL;  BestR = R;
6:    end if
7:    return;
8: end if
9: if d > N then {Continue with next source node}
10:   s++; d = 1;
11:   SRRS(s, d, R, LB);
12: end if
13: if s = d then {Continue with next destination node}
14:   d = s + 1;
15:   SRRS(s, d, R, LB);
16: end if
17: //Main Recursion
18: for k = 1; k ≤ |K_sd|; k++ do
19:   newR = add path p^k_sd to R;   //Update R
20:   newLB = update LB as described in the text;
21:   if newLB < BestSOL then
22:      SRRS(s, d + 1, newR, newLB);
23:   else
24:      return;
25:   end if
26: end for
```

Whenever a path between a source-destination pair is added to form a new partial configuration, the lower bound for the new configuration is calculated by updating the spectrum slots along the links of the new path, as discussed above. If this lower bound is not better than the best current solution, then the recursion terminates immediately: all complete routing configurations that may be derived from this partial configuration are guaranteed to have a solution that is worse than the current one, hence there is no need to generate or evaluate them.

To describe the SRRS recursion, we may think of a routing configuration R as a $N \times N$ matrix. If R is a complete configuration, then element[5] $R[sd], s \neq d$, is one of the paths from s to d, i.e., $R[sd] \in K_{sd}, s \neq d$. If R is a partial configuration, then some elements $R[sd], s \neq d$ may be null,

[5]We assume that there is no traffic from a node to itself, hence elements $R[sd]$ are always null for $s = d$.

i.e., the corresponding paths have yet to be filled in. We also assume that there are two global variables, $BestSOL$ and $BestR$, that store the best solution and corresponding best complete configuration, respectively, that have been found so far. We initialize $BestR$ to the configuration with $R[sd] = p_{sd}^1$ for all node pairs (s, d), and $BestSOL$ to the solution obtained by applying the LFF algorithm to this configuration. For instance, this configuration and solution may correspond to shortest path routing.

The first call is to $SRRS(s = 1, d = 2, R_0, 0)$, where R_0 is the empty configuration (no paths) and the corresponding lower bound $LB(R_0) = 0$. There are three base cases for the recursion.

1) Whenever $s > N$ (Lines 1-9), the algorithm has considered all possible node-pairs and has produced a complete configuration R. This configuration is evaluated by applying the LFF algorithm to perform spectrum assignment. If the result is better than the best known solution so far, the global variables $BestSOL$ and $BestR$ are updated accordingly. The recursion terminates at that point.
2) If $d > N$ (Lines 10-13), the algorithm has considered all node pairs with source nodes up to and including s, and has produced a corresponding partial configuration. It then makes a recursive call to consider node pairs with source $s + 1$ (and beyond).
3) Finally, if $s = d$ (Lines 14-17), since there are no paths between a node and itself to consider, the algorithm makes a recursive call to continue with the next destination $d = s + 1$.

The main body of the algorithm is the **for** loop in Lines 19-27. This loop iterates over all possible paths between the given node pair (s, d). For each path p_{sd}^k, it installs it in the partial routing configuration passed to this call (which did not include such a path) and updates the lower bound as we discussed earlier. If the new lower bound is less than the best known solution, then this is a promising new configuration that has the potential to lead to a better solution. Therefore, the algorithm makes a recursive call to consider the next destination. On the other hand, if the new lower bound is equal to or higher than the best known solution, then adding paths (and hence traffic demands to the corresponding links) is guaranteed to produce solutions that are no better than the current best one. In this case, no recursive call is made, eliminating this part of the search space from further consideration.

To illustrate, consider the initial call to $SRRS(s = 1, d = 2, R_0, 0)$. The **for** loop of this call will make $K_{1,2}$ recursive calls, each of which will have as argument a routing configuration with only one path, i.e., one of the $K_{1,2}$ paths $p_{1,2}^k$ between nodes 1 and 2. In turn, each of these calls will make $K_{1,3}$ recursive calls, each of which will have a configuration with only two paths, the path $p_{1,2}^k$ passed to it and one of the $K_{1,3}$ paths between nodes 1 and 3. Each of these recursions will continue until either (1) a complete configuration is reached (first base case) that contains one path for every source-destination node, or (2) the lower bound of an intermediate partial configuration becomes higher than the current best solution, at which time the corresponding recursion ends.

We emphasize that, in the worst case, the SRRS recursion may be forced to generate all, or close to all, possible routing configurations and hence take exponential time to complete. However, the simulation results we present in the next section indicate that, in practice, the SRRS algorithm will need to explore only a tiny fraction of the routing space. Thus, SRRS represents a scalable solution to large RSA problems.

V. Simulation Study

We have evaluated the performance of the SRRS algorithm by carrying out simulation experiments with RSA problem instances characterized by three parameters: (1) the network topology, (2) the set K_{sd} of paths for each source-destination pair (s, d), and (3) the probability distribution of traffic demands. We considered two network topologies, the 14-node, 21-link NSFNet and the 32-node, 54-link GEANT2 network. We used DFS to generate the $|K_{sd}| = k$ shortest paths for each source-destination pair, $k = 1, 2, 3$, as well as *all* paths between each node pair; we denote the latter as $k = all$. We assume symmetric routing of demands, i.e., traffic from s to d takes the same path as traffic from d to s, for all s, d.

We assume data rates of 10, 40, 100, 400, and 1000 Gbps. For a given instance, we generate a random value for the demand between a pair of nodes based on one of three distributions: 1) *Uniform*: each of the five rates is selected with equal probability; 2) *Skewed low*: the rates above are selected with probability $0.30, 0.25, 0.20, 0.15$, and 0.10, respectively; or 3) *Skewed high*: the five rates are selected with probability $0.10, 0.15, 0.20, 0.25$, and 0.30, respectively.

The metric we consider is the maximum number of spectrum slots on any network link that SRRS returns as the (best) solution value SOL. We let SRRS$_k$ denote the version of the algorithm that generates k paths for each node pair in Step 1, $k = 1, 2, 3, all$, and let SOL_k denote the corresponding solution. Since SRRS uses the LFF heuristic for spectrum allocation, it is important to compute a lower bound in order to evaluate the quality of a solution SOL_k. Consider a complete routing configuration R generated by SRRS within the **if** statement of Lines 1-9. The lower bound $LB(R)$ given by expression (2) on this configuration is a lower bound on the optimal solution of this RSA instance. Therefore, we modify the algorithm to calculate not only the LFF solution on R in Line 3, but also the lower bound $LB(R)$. Then, the lower bound LB_k for algorithm SRRS$_k$ is taken as the minimum of the lower bounds $LB(R)$ over all complete configurations generated by this algorithm. Finally, the overall lower bound is: $LB = \min_k\{LB_k\}$.

The metric we use to characterize the quality of the solution constructed by algorithm SRRS$_k$ is the ratio $Q_k = SOL_k/LB$. Clearly, $Q_k \geq 1$; the closer Q_k is to 1, the better the solution, i.e., the closer it is to the optimal one.

Figures 1 plot the average quality ratio Q_k for the NSFNet and GEANT2 topologies. Each figure includes three curves, one each for demand matrices generated by the uniform, skewed low, and skewed high distributions, respectively. Each

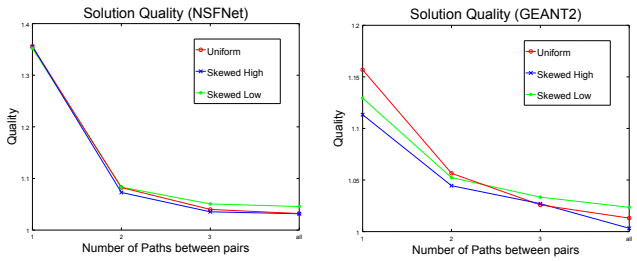

Fig. 1. Average ratio, Q, vs the number k of paths, NSFNet and GEANT2

TABLE I
RUNNING TIME (SEC) OF SRRS FOR NSFNET AND GEANT2

	K	Uniform	Skewed high	Skewed low
NSFNet	$k=2$	1,222	279	1,381
	$k=3$	1,513	2,231	2,614
	$k=all$	5,183	5,919	3,398
GEANT2	$k=2$	12,102	13,294	13,668
	$k=3$	12,799	13,661	13,391
	$k=all$	11,174	10,715	12,204

data point is the average of 100 random problem instances generated for the stated parameters (i.e., network topology, number k of paths, and traffic demand distribution).

As we can see, the solution quality improves (i.e., the ratio Q_k decreases) with the number k of paths. Note that $k = 1$ in the figures corresponds to spectrum assignment along the shortest paths (i.e., a single routing configuration), while as k increases the algorithm considers a significantly larger set of routing configurations and hence is able to find increasingly better solutions. This improvement in solution quality as k increases is observed for both topologies and across all three traffic distributions. For the smaller NSFNet topology, we also observe that with as few as $k = 3$ paths most benefits of the routing search are realized, and there is little improvement even when we consider all possible paths between each node pair. For the larger GEANT2 network, on the other hand, there is a substantial improvement in solution quality as we move from $k = 3$ to all paths. Finally, we note that, while SRRS searches the entire routing space, it uses a heuristic spectrum assignment algorithm. As a result, the SRRS algorithm may not obtain the optimal solution even when it is given as input all the paths between each node pair. On the other hand, we also emphasize that the lower bound may not be achievable, and hence, the ratio Q_k for optimal solution may be strictly larger than 1.0. In any case, we observe that the solutions obtained for all paths achieve a ratio that is very close to 1.0.

Table I lists the running time (in seconds) of the SRRS algorithm for the NSFNet and GEANT2 networks, averaged over the 100 problem instances for the stated topology, number k of paths, and traffic distribution. We run SRRS on the Aziz Supercomputer facility at King Abdulaziz University. Each problem instance was run on *a single core* (i.e., no parallelization). To appreciate the effectiveness of SRRS, we note that for a network with N nodes and k paths per node, there are $O(k^{N(N-1)/2})$ routing configurations under the symmetric routing we consider; i.e., just for $k = 2$ paths, the size of the routing space is $O(2^{91})$ for NSFNet and $O(2^{496})$ for GEANT2. Nevertheless, SRRS eliminated huge swaths of the routing space and completed in less than two hours for NSFNet and less than four hours for GEANT2. In other words, the algorithm scales well to problem instances representative of topologies encountered in practice.

VI. CONCLUDING REMARKS

In this work we have made two contributions: (1) a new decomposition of network optimization problems that completely and exactly decouples the routing aspect from the resource allocation aspect, and (2) a new recursive branch-and-bound method that searches the entire routing space efficiently. Our current efforts are directed to developing a recursive algorithm for generic resource allocation problems, and to adapting SRRS for parallel execution on multi-core architectures.

REFERENCES

[1] J. M. Simmons, *Optical network design and planning*. Springer, 2014.
[2] J. Simmons and G. N. Rouskas, "Routing and wavelength (spectrum) allocation," in *Springer Handbook of Optical Networks*, 2020.
[3] G. N. Rouskas, "Routing and wavelength assignment in optical wdm networks," *Wiley Encyclopedia of Telecommunications*, 2003.
[4] R. Dutta, G. N. Rouskas, *et al.*, "A survey of virtual topology design algorithms for wavelength routed optical networks," *Optical Networks Magazine*, vol. 1, no. 1, pp. 73–89, 2000.
[5] B. Jaumard, C. Meyer, and B. Thiongane, "Comparison of ilp formulations for the rwa problem," *OSN*, vol. 4, no. 3-4, pp. 157–172, 2007.
[6] R. Dutta and G. N. Rouskas, "Traffic grooming in wdm networks: Past and future," *IEEE network*, vol. 16, no. 6, pp. 46–56, 2002.
[7] H. Wang and G. N. Rouskas, "Hierarchical traffic grooming: A tutorial," *Computer Networks*, vol. 69, pp. 147–156, 2014.
[8] M. Klinkowski, P. Lechowicz, and K. Walkowiak, "Survey of resource allocation schemes and algorithms in spectrally-spatially flexible optical networking," *OSN*, vol. 27, no. C, pp. 58–78, 2018.
[9] S. Talebi, F. Alam, I. Katib, M. Khamis, R. Salama, and G. N. Rouskas, "Spectrum management techniques for elastic optical networks: A survey," *Optical Switching and Networking*, vol. 13, pp. 34–48, 2014.
[10] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE network*, vol. 14, no. 6, pp. 16–23, 2000.
[11] B. Jaumard and M. Daryalal, "Efficient spectrum utilization in large scale rwa problems," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 2, pp. 1263–1278, 2017.
[12] Z. Liu and G. N. Rouskas, "Link selection algorithms for link-based ILPs and applications to RWA in mesh networks," in *ONDM*, 2013.
[13] Z. Liu and G. N. Rouskas, "A fast path-based ILP formulation for offline RWA in mesh optical networks," in *GLOBECOM*, pp. 2990–2995, 2012.
[14] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 29. wh freeman New York, 2002.
[15] K. Christodoulopoulos, I. Tomkos, and E. Varvarigos, "Routing and spectrum allocation in OFDM-based optical networks with elastic bandwidth allocation," in *Proceedings of IEEE Globecom*, 2010.
[16] B. Jaumard, C. Meyer, and B. Thiongane, "On column generation formulations for the rwa problem," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1291–1308, 2009.
[17] M. Dawande, R. Gupta, S. Naranpanawe, and C. Sriskandarajah, "A traffic-grooming algorithm for wavelength-routed optical networks," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 565–574, 2007.
[18] J. Enoch and B. Jaumard, "Towards optimal and scalable solution for routing and spectrum allocation," *Electronic Notes in Discrete Mathematics*, vol. 64, pp. 335–344, 2018.
[19] M. Klinkowski, M. Żotkiewicz, K. Walkowiak, M. Pióro, M. Ruiz, and L. Velasco, "Solving large instances of the rsa problem in flexgrid elastic optical networks," *IEEE/OSA JOCN*, vol. 8, no. 5, pp. 320–330, 2016.
[20] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas, "Spectrum assignment in optical networks: A multiprocessor scheduling perspective," *IEEE/OSA JOCN*, vol. 6, no. 8, pp. 754–763, 2014.
[21] M. Fayez, I. Katib, G. N. Rouskas, and H. Faheem, "Spectrum assignment in mesh elastic optical networks," in *ICCCN*, 2015.