Parameterized First Fit (PFF): Eliminating Symmetry in Spectrum Allocation

George N. Rouskas, Priya Sharma, Shubham Gupta
North Carolina State University

***Abstract*—Spectrum allocation (SA) is a fundamental problem in optical network design, yet existing solutions cannot cope effectively with the challenges posed by spectrum symmetry. In this work, we develop parameterized first-fit (PFF), a new heuristic for the SA problem that is not affected by spectrum symmetry and has several desirable properties: it explores a pre-defined subset of the solution space whose size is tailored to the available computational budget; it constructs this subset by sampling from diverse areas of the solution space rather than from the neighborhood of an initial solution; it finds solutions by applying the well-known FF heuristic and thus it can be deployed readily; and it is efficient in finding good quality solutions.**

I. Introduction

The design and planning of optical networks encompasses the allocation of optical spectrum resources to traffic demands as an integral part of the optimization process [1]. Spectrum allocation (SA) [2], a generalization of wavelength allocation (WA) [2], [3], is tightly coupled to other aspects of network design, including the routing process [4]–[6], traffic grooming [7], virtual topology embedding [8], [9], and network survivability [10]. Therefore, since the early days of optical networking, researchers and industry practitioners have focused on developing effective spectrum allocation strategies. These efforts, however, have been complicated by two features inherent to the SA and WA problems: spectrum continuity and spectrum symmetry.

Due to the spectrum continuity property of optical elements, a connection that optically bypasses a node must exit on the same optical frequency it entered. Hence, the spectrum resources that are in use on one link may affect the resources that may be allocated on other links, creating resource contention among the links of the network. As a result, the SA problem is computationally intractable in general topologies [11], even when it is not coupled to other objectives (e.g., routing).

Symmetry refers to the fact that spectrum slots are *interchangeable* [12]. Hence, for each possible solution, a large number of equivalent solutions may be derived simply by using a different permutation of spectrum slots [13]. Symmetry is particularly challenging for conventional ILP formulations of the SA problem, regardless of whether these were developed specifically for SA or as part of formulations that tackle the more general routing and spectrum allocation (RSA) problem. Since an ILP solver will have to evaluate an exponential number of distinct but equivalent optimal solutions, its running time can be unnecessarily long [13]. ILP formulations based on maximal independent sets (MIS), such as the one we developed in [14] for the RWA problem in rings, do not suffer from symmetry. However, MIS-based formulations are impractical for general topology networks as the number of variables increases exponentially with the network size.

Given these two challenges, the SA problem is typically solved using heuristic algorithms that attempt to minimize spectrum contention. These include the first-fit, best-fit, most-used, and least-loaded heuristics [15], each representing a different tradeoff between algorithmic complexity and amount of network state information required. In particular, first-fit (FF) is a simple heuristic that operates without any global knowledge, performs well across various network topologies and traffic demands [2], [16], and, consequently, it is commonly employed for spectrum/wavelength allocation.

In recent work [17] we proved that there exists a permutation of the traffic demands such that applying the FF heuristic to this permutation yields an optimal solution for the SA problem. Based on this optimality property, we developed recursive first-fit (RFF), an optimal branch-and-bound algorithm. RFF searches the entire space of demand permutations to find one that is optimal for the SA problem at hand, and applies the FF heuristic as it incrementally builds each permutation during the search. While the demand permutation space is itself exponential in size, RFF represents a significant improvement over existing approaches as it completely sidesteps the spectrum symmetry challenge.

Even so, it is impossible to explore the demand permutation space for networks of realistic size. Therefore, in this work we develop a heuristic for the SA problem that has three desirable properties: 1) it employs an intuitive parameter to construct a subset of the demand permutation space to explore with a size that matches the available computational resources; 2) it selects permutations that are distributed uniformly across the permutation space (i.e., they are not limited to any particular region of the space); and 3) it can be readily deployed as it simply applies the well-known FF heuristic.

In Section II we discuss the SA problem we consider, explain the concept of spectrum symmetry, and show how an optimality property of the FF heuristic allows for the elimination of symmetric solutions and leads to a vast reduction of the solution space to be explored. In Section III we present parameterized FF (PFF), a heuristic for the SA problem that allows the network designer to explore cutomizable subsets of permutations that sample from diverse regions of the solution space. We evaluate the PFF algorithm in Section IV, and we

The Offline SA Problem
Input:

- A graph $G = (V, A)$
- A set $\mathcal{T} = \{T_i = (s_i, d_i, p_i, t_i), i = 1, \cdots, K\}$, of traffic requests

Output: An assignment of t_i spectrum slots to each request T_i along the physical path p_i
Spectrum Constraints:

- *Contiguity:* each request T_i is allocated a block of t_i contiguous spectrum slots
- *Continuity:* each request is allocated the same block of spectrum slots along all links of its path p_i
- *Nonoverlap:* requests whose paths share a link are allocated nonoverlapping blocks of spectrum slots

Objective: Minimize the index of the highest spectrum slot used on any link in the network

Fig. 1. The Offline SA problem

conclude the paper in Section V.

II. The SA Problem, Spectrum Symmetry, and the First-Fit Optimality Property

Consider an optical network with topology graph $G = (V, A)$, where V is the set of nodes and A is the set of directed fiber links in the network. Let $N = |V|$ denote the number of nodes and $L = |A|$ the number of directed links. The traffic offered to the network consists of a set $\mathcal{T} = \{T_i\}$ of K traffic requests. Each request is a tuple $T_i = (s_i, d_i, p_i, t_i)$, where: s_i is the source and d_i the destination node of the request; p_i is the path between nodes s_i and d_i that the request must follow; and t_i is the number of spectrum slots required to carry the traffic from s_i to d_i.

In this work we study the offline SA problem shown in Figure 1, where the objective is to allocate spectrum slots to each traffic request so as to minimize the highest assigned slot on any link, while satisfying the three spectrum constraints. This objective attempts to pack the spectrum slots assigned to the traffic requests as tightly as possible, and hence it minimizes spectrum fragmentation and allows for growth in demand; consequently, it is one that has been adopted widely in the literature. Also, we assume that the path p_i of each request T_i is fixed and pre-determined, i.e., any routing decision has been made before the allocation of spectrum. Therefore, any algorithm that solves this SA problem, including the one we propose in the next section, may be applied as part of a multistep, iterative approach to the RSA problem [2].

We have shown [11] that the SA problem is NP-hard even for chain (i.e., single-path) networks with four or more links. But even beyond computational intractability, a major challenge in tackling the SA problem in Figure 1, or any of its variants that have been studied in the literature, relates to *spectrum symmetry*. Specifically, blocks of contiguous spectrum slots of a certain size are interchangeable. Therefore, for any optimal solution to the SA problem, one can derive a large number of equivalent solutions simply by permuting the spectrum blocks.

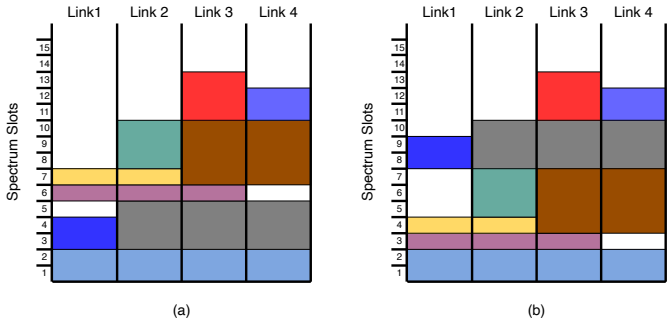

Fig. 2. Equivalent solutions due to spectrum symmetry

Figure 2 illustrates how spectrum symmetry leads to multiple equivalent solutions. Figure 2(a) shows a solution to the SA problem on a four-link chain network with $K = 9$ requests. Each request is represented by a different color and spans all the links in the path of the corresponding demand. For instance, the bottommost (light blue) request spans all four links of the network, indicating that the request has been allocated this contiguous block of two spectrum slots along each of these links. Note that the solution shown in Figure 2(a) is optimal: the highest assigned slot on Link 3 is equal to the lower bound, i.e., the number of slots required to carry the traffic requests whose path includes Link 3. Consider now two blocks of spectrum slots in Figure 2(a): the three-slot block consisting of specrum slots 3-5, and the five-slot block consisting of spectrum slots 6-10. Figure 2(b) shows the equivalent solution that can be obtained by permuting these two blocks of slots. In the new solution, the two requests that were allocated slots in the range 3-5 in Figure 2(a) are now shifted up and are allocated the corresponding slots in the range 8-10, while the four requests that were allocated slots in the range 6-10, are now shifted down accordingly. Otherwise, the two solutions in Figures 2(a) and (b) are identical; they are also equivalent in that they yield the same objective function value. Furthermore, note that 1) it is possible to obtain many more solutions equivalent to the two shown in Figure 2 by permuting different blocks of spectrum slots, and 2) spectrum symmetry applies to non-optimal solutions as well.

Based on the above discussion it is clear that, due to spectrum symmetry, conventional ILP formulations may yield an exponentially large number of equivalent (optimal or sub-optimal) solutions. Consequently, ILP solvers are forced to explore a solution space that is essentially the product of the request permutation space and the spectrum permutation space. However, exploring such a vast solution space is unnecessary.

Let us return to the offline SA problem shown in Figure 1 on graph G and request set $\mathcal{T} = \{T_i, i = 1, \cdots, K\}$. Let P be a permutation (i.e., an ordering) of the traffic requests T_i. Let $SOL(P)$ denote the solution to the SA problem obtained by the FF heuristic when it considers each traffic request in

the order implied by permutation P. Let OPT denote the objective value of an optimal solution to the SA problem. Clearly, for any permutation P of the traffic requests it must be that $OPT \leq SOL(P)$.

We have shown in [17] that there exists a permutation $P^{\star}_{FF}$ of the traffic requests such that applying the FF heuristic to the requests in the order in which they appear in $P^{\star}_{FF}$ yields an optimal solution to the SA problem, i.e., $SOL(P^{\star}_{FF}) = OPT$. This optimality property of FF implies that, to find an optimal solution, it is sufficient to examine all permutations of the traffic demands and, therefore, there is no need to consider a spectrum assignment for the various request permutations other than the one produced by the FF heuristic. For instance, it is not difficult to see that the solution shown in Figure 2(a) is the product of the FF heuristic on an appropriate permutation of the $K = 9$ traffic requests. However, the equivalent solution in Figure 2(b) would not be produced by the FF heuristic; rather, FF would have allocated slots 5 and 6 to the (dark blue) request spanning just Link 1, not slots 8 and 9.

Although the number of traffic request permutations is exponential, the FF optimality property allows us to design algorithms that ignore the exponential number of symmetric solutions derived from spectrum permutations, e.g., solutions such as the one in Figure 2(b). Doing so, drastically reduces the size of the solution space that needs to be explored. Accordingly, we developed recursive first-fit (RFF), a branch-and-bound algorithm that recursively searches the entire space of demand permutations to find an optimal solution [17].

This FF optimality property explains why many studies of the SA (and WA) problem have confirmed that the FF heuristic yields good solutions across diverse problem instances. It also suggests a procedure for finding the unknown permutation $P^{\star}_{FF}$: enumerate all request permutations of requests and select the one for which the FF heuristic yields the smallest objective value. However, in a network with N nodes and traffic between all node pairs, the size K of set $\mathcal{T}$ is $O(N^2)$. Therefore, any algorithm, such as RFF, that considers all possible permutations of requests to determine the optimal spectrum allocation must take time that is exponential in the size of the network, $O(N^2!)$. Next, we present a parameterized heuristic that applies the FF algorithm to a subset of the request permutation space whose size can be customized to the available computational budget.

III. Parameterized First Fit (PFF)

A. Motivation

The motivation for a new algorithm for the SA problem is based on the observation that the FF heuristic and the optimal RFF algorithm we developed in [17] represent two opposite extremes in exploring the solution space of request permutations. The FF heuristic considers a single permutation, whereas, given sufficient time, RFF will explore all $K!$ permutations. While RFF can be executed in parallel [17], exploring the entire permutation space for SA instances encountered in practice would be infeasible. Typically, there is a budget in terms of the computational resources (or time) alloted to tackling network design problems such as SA. Therefore, any algorithm, either optimal or heuristic in nature, that runs for a limited amount of time is bound to explore only the region of the solution space around its starting point, and hence fail to find optimal solutions that may exist in different parts of the space.

Furthermore, algorithms that operate in a branch-and-bound fashion, including RFF and those employed by integer linear programming (ILP) solvers, are sensitive to the values of the input parameters, in this case the spectrum demands t_i. Consequently, given two different problem instances on the same topology graph $G = (V, A)$ and number of requests K, branch-and-bound algorithms will explore a different number of permutations (i.e., a different fraction of the solution space) for each instance within a prescribed amount of running time. In addition, it cannot be known *a priori* for which instance the algorithm will explore a larger or smaller fraction of the solution space; and it may be difficult to determine the relative size of the solution space explored for each instance after the algorithm has completed.

Parameterized first fit, PFF(M), where M is a parameter such that $1 \leq M \leq K$, is a generalization of the FF and RFF algorithms that operates in the vast space between these two extremes. Rather than searching the whole solution space of size $O(K!)$ from some initial and often arbitrary starting point, the key idea of PFF(M) is to completely explore a subset of the solution space of size $O(M!)$, where $M \leq K$, and, typically, $M \ll K$. Ideally, the $M!$ request permutations to be explored should be distributed evenly across the whole solution space. One strategy for achieving this goal would be to generate the $M!$ permutations randomly. Instead, PFF(M) takes a structured approach to generating the $M!$ permutations that has several benefits for network designers:

- it provides a well-defined tradeoff between the size of the permutation space to be explored and the amount of computational resources available, via the choice of the value of parameter M;
- it employs a method that is readily reproducible, works for every value of parameter M, and explores the exact same subset of the solution space for any two instances of the same SA problem,
- it explores request permutations that are spread over diverse regions of the solution space, and
- it provides better and more even coverage of the solution space as the value of M increases.

B. The PFF Algorithm

In determining a solution to an instance of the SA problem, PFF(M) considers only subsets of the solution space of size equal to that of a (smaller) permutation space, i.e., of size $O(M!), M \leq K$. It then generates $M!$ request permutations that are spread over the entire original solution space.

To achieve this objective, PFF(M) first partitions the set $\mathcal{T}$ of K requests into $M \leq K$ subsets, $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_M$, and

TABLE I
PFF PERMUTATION SPACE FOR THE SET $\mathcal{T} = \{A, B, C, D, E, F, G\}$ OF $K = 7$ REQUESTS, PARTIONED INTO $M = 3$ SUBSETS, $\mathcal{T}_1 = \{A, B, C\}, \mathcal{T}_2 = \{D, E\}, \mathcal{T}_3 = \{F, G\}$

Subset permutations	Request permutations
$\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$	A, B, C, D, E, F, G
$\mathcal{T}_1, \mathcal{T}_3, \mathcal{T}_2$	A, B, C, F, G, D, E
$\mathcal{T}_2, \mathcal{T}_1, \mathcal{T}_3$	D, E, A, B, C, F, G
$\mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_1$	D, E, F, G, A, B, C
$\mathcal{T}_3, \mathcal{T}_1, \mathcal{T}_2$	F, G, A, B, C, D, E
$\mathcal{T}_3, \mathcal{T}_2, \mathcal{T}_1$	F, G, D, E, A, B, C

generates all $M!$ permutations of the M subsets. We assume that the requests within each subset are listed in a fixed order, and we consider this list as a single *meta-request*. In essence, then, this operation produces all $M!$ meta-request permutations.

PFF(M) then generates the $M!$ *request* permutations to evaluate by parsing each meta-request permutation and replacing each meta-request with the list of its constituent requests. This approach produces request permutations that are spread over the entire solutions space. To see this, note that two meta-request permutations that differ in the first meta-request will produce two request permutations that are far apart in the request permutation space; similar observations apply to permutations that differ in the second, third, etc., meta-request. In other words, replacing each meta-request with its individual requests involves large jumps in the request permutation space, achieving our objective of generating request permutations that cover the entire solution space.

Finally, PFF applies the FF heuristic to the $M!$ request permutations created in this manner, and selects the permutation that results in the best solution to the SA problem.

Table I illustrates this concept for the set $\mathcal{T} = \{A, B, C, D, E, F, G\}$ of $K = 7$ requests partitioned into three subsets, $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}_3$. There are several options for partitioning the set $\mathcal{T}$ into subsets. In this work, we only consider partitions in which the sizes of the various subsets vary by at most one. Therefore, each subset $\mathcal{T}_i$ is such that $|\mathcal{T}_i| = \lfloor K/M \rfloor$ or $\lfloor K/M \rfloor + 1$. Without loss of generality, we determine the subsets such that $|\mathcal{T}_1| \geq |\mathcal{T}_2| \geq \cdots \geq |\mathcal{T}_M|$. Therefore, as shown in Table I, the $M = 3$ subsets of $\mathcal{T}$ are: $\mathcal{T}_1 = \{A, B, C\}$, $\mathcal{T}_2 = \{D, E\}$, and $\mathcal{T}_3 = \{F, G\}$.

The left column of Table I shows the $M! = 6$ subset permutations (or meta-request permutations, as we mentioned above). The right column of the tabe shows the corresponding 6 request permutations obtained by replacing each subset (met-request) with its constituent requests. Although this is a small number compared to the $7! = 5,040$ possible request permutations to provide uniform coverage of the solution space, it is evident that these six permutations are spread across different regions of the space. In this example, PFF will evaluate only the 6 permutations shown in the right column of the table by running the FF heuristic on each.

Algorithm 1 shows the operation of PFF(M). The preprocessing step in Lines 1-6 generates the M subsets of the request set $\mathcal{T}$, and from them the $M!$ request permutations that the algorithm considers. This step takes time $O(M!)$, but since M is fixed (i.e., it is determined by the network designer and is not part of the input to the problem), this time can be considered as constant. Note that a network designer may have to solve multiple instances for a given SA problem defined by the network topology $G = (V, A)$ and number of spectrum requests K; for instance, this may be due to carrying out a "what-if" analysis to explore the sensitivity of design decisions to forecast traffic demands. In this case, the designer only needs to perform the preprocessing step once, store the $M!$ permutations, and use them to solve all instances that are part of the analysis. Therefore, the computational cost of this step can be amortized over multiple problem instances.

The main part of the algorithm in Lines 7-13 simply runs the FF heuristic on each of the $M!$ generated in the preprocessing step, and selects the one that offers the best solution to the problem at hand. Each application of the FF heuristic takes time $O(KL)$, as each permutation consists of K requests and each request may involve any of the L links in the network. Therefore, the total running time of this part of the algorithm is $O(KLM!)$, where M is again considered a constant.

The PFF(M) algorithm encompasses the FF heuristic (for $M = 1$) and the optimal RFF algorithm (for $M = K$), as special cases. Parameter M affords the network designer a wide

Algorithm 1 Parameterized First Fit

Input:
$G = (V, A)$: network topology
$\mathcal{T} = \{T_i = (s_i, d_i, p_i, t_i\}$: set of traffic requests
$K = |\mathcal{T}|$: number of traffic requests

Output:
$BestP$: best request permutation
$BestSOL$: SA solution corresponding to $BestP$

PFF$(M, 1 \leq M \leq K)$
1: {**Preprocessing:** Generate $M!$ request permutations}
2: Partition $\mathcal{T}$ into M subsets, $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_M$, such that $|\mathcal{T}_i| = \lfloor K/M \rfloor$ or $\lfloor K/M \rfloor + 1$;
3: Generate all $M!$ permutations of the M subsets;
4: **for** $i = 1; i \leq M!; i$++ **do**
5: $P_i \leftarrow$ request permutation created by replacing each subset in subset permutation i with its requests;
6: **end for**
7: {**Main:** Apply FF to the $M!$ request permutations}
8: **for** $i = 1; i \leq M!; i$++ **do**
9: $S \leftarrow SOL(P_i)$; {solution obtained by FF on P_i}
10: **if** $S < BestSOL$ **then**
11: $BestSOL = S$; $BestP = P_i$;
12: **end if**
13: **end for**
14: return;

range of options between these two extremes, and its value can be selected so as to strike an appropriate balance between the quality of solution and the running time $O(KLM!)$ of the FF application step of the algorithm.

As a final note, let $SOL_{PFF}(M)$ denote the value of the best solution returned by PFF(M). Increasing the value of parameter M from, say, m to $m+1$, results in an increase in the size of the solution space explored by a factor of $m+1$. Nevertheless, despite the fact that PFF($m+1$) evaluates a larger number of request permutations than PFF(m), it does *not* necessarily follow that $SOL_{PFF}(m+1) \leq SOL_{PFF}(m)$. Intuitively, since PFF($m+1$) and PFF(m) each evaluate a different set of request permutations, it is conceivable that a permutation in the set of PFF(m) may yield a solution that is better than those produced by permutations in the set of PFF($m+1$). Our experiments confirm this observation as we have occasionally found that $SOL_{PFF}(m) < SOL_{PFF}(m+1)$ on the same problem instance. Therefore, in this work we apply the PFF(M) algorithm as follows:

1) Run PFF(m) for all $m = 1, \cdots, M$.
2) Return $SOL_{PFF} = \min_{m=1,\cdots,M}\{SOL_{PFF}(m)\}$.

Consequently, the solutions in Step 2 above are monotonically non-increasing as a function of m.

IV. Simulation Study

In our simulation study to evaluate the performance of the PFF heuristic we considered three algorithms: 1) **First-Fit heuristic, FF:** We run FF on a permutation in which the K traffic requests are listed in decreasing order of spectrum demand t_i, and requests with the same demand are listed in decreasing order of path length. 2) **Recursive FF, RFF:** This is the branch-and-bound algorithm we developed in [17]. We run RFF until it either reaches the lower bound (in which case it has found an optimal solution) or it reaches a 5-hour limit on running time (in which the solution may not be optimal). 3) **Parameterized FF, PFF(M):** In our experiments, for parameter M we used the values $M = 1, \cdots, 6$. Note that PFF(1) is equivalent to FF, and PFF(K) is equivalent to RFF, where K is the number of input traffic requests.

We consider the highest index of allocated spectrum slots on any network link as the performance measure to compare the various algorithms. For a meaningful comparison between different problem instances, we normalize the solutions returned by the various algorithms by dividing with the lower bound for the corresponding instance. A lower bound LB on the optimal objective value may be obtained by ignoring the spectrum contiguity and continuity constraints and simply counting the spectrum slots required by all traffic demands on the most congested link: $LB = \max_{l \in A}\left\{\sum_{T_i \in \mathcal{T}: l \in p_i} t_i\right\}$. Clearly, the closer the normalized value is to 1.0, the better the solution.

To be consistent with our recent work in [17], we create SA problem instances by generating traffic requests between all node pairs in the network as follows. We consider data rates of 10, 40, 100, 400, and 1000 Gbps. For a given problem 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. Once the traffic rates between each node pair have been generated, we calculate the corresponding spectrum slots by assuming that the slot width is 12.5 GHz, and adopting the parameters of [18]–[21] to determine the number of spectrum slots that each demand requires based on its data rate and path length. For each traffic distribution, we generate 100 random problem instances. We also use shortest path routing to find a path for each traffic request before we apply one of the SA algorithms above.

Table II summarizes the results we have obtained regarding the relative performance of the FF, PFF(M), and RFF algorithms on the well-known 14-node, 21-link NSFNET topology. Specifically, the table shows how far the solutions obtained by each algorithm are from the lower bound; the values listed in the table are averages over the 100 problem instances from the specified traffic distribution. The PFF(M) values were calculated by considering the monotonically non-increasing solutions we discussed at the end of the previous section.

As we can see, the FF algorithm performs well and, on average, produces solutions that are within 9-10% of the lower bound across the 300 problem instances we used in our experiments. These results are consistent with earlier research indicating that the FF algorithm finds solutions of good quality. Also, similar to what we have shown in [17], the branch-and-bound RFF algorithm improves upon the FF heuristic and finds solutions that, on average, are between 4-6% from the lower bound, depending on the traffic distribution.

Turning our attention to the PFF(M) results in Table II, we make several observations. The quality of solutions improves as a function of M, a fact that is consistent with our use of monotonically non-increasing solutions. On average, the improvement over the FF solutions is substantial even for small values of M; for instance, using $M = 2$ only takes an additional fraction of a second (refer to Table III we discuss shortly) and leads to an improvement of about 2%. On the other hand, increasing the value of M results in diminishing returns, and by $M = 6$ most of the benefits of this approach have been realized. This outcome is expected because 1) the FF solutions are of high quality to begin with, 2) the PFF(6) solutions are very close to the lower bound on average, and 3) the lower bound may not be attainable in general.

TABLE III
Running Time (seconds) of FF, RFF and PFF(M) algorithms, NSFNet topology

FF	PFF(M)					RFF
[PFF(1)]	$M = 2$	3	4	5	$M = 6$	[PFF(91)]
0.24	.49	1.35	5.16	28.54	171.28	$\leq 18,000$

The most important observation from Table II, however, has

TABLE II
RELATIVE PERFORMANCE OF FF, RFF AND PFF(M) ALGORITHMS AS % FROM LOWER BOUND, NSFNET TOPOLOGY

Traffic Distribution	FF [PFF(1)]	PFF(M) $M=2$	$M=3$	$M=4$	$M=5$	$M=6$	RFF [PFF(91)]
Skewed High	9.04%	6.89%	4.91%	3.77%	3.30%	3.16%	3.74%
Skewed Low	10.29%	8.31%	7.42%	7.00%	6.74%	6.70%	6.01%
Uniform	9.09%	6.29%	4.73%	4.18%	4.06%	4.01%	4.49%

to do with the fact that the PFF algorithm is not just better than FF but that it can find solutions that are even better than the ones obtained by the optimal branch-and-bound RFF algorithm after five hours of execution. Specifically, for the skewed high distribution, PFF(M) almost matches the RFF performance (on average) starting with $M = 4$ and outperforms it starting with $M = 5$; while for the uniform distribution, PFF(M) outperforms RFF (on average) starting with $M = 4$. It is only for the skewed low distribution that RFF produces better solutions than PFF for the values of M we consider here.

To put these results into perspective, we have listed in Table III the running time of the algorithms; the running time of PFF does not depend on the demand distribution, hence the values shown are representative of all three distributions. The 14-node NSFNet has 91 node pairs, and each problem instance we solve consists of $K = 91$ traffic requests (hence RFF is denoted as PFF(91) in the tables). Therefore, the size of the demand permutation space is $O(91!)$. While RFF explores tens of millions of permutations in five hours [17], PFF(M) only explores $M!$ permutations and for $M = 6$ takes less than 200 seconds; the running time of PFF(M) is roughly M times that of PFF($M-1$), as the former examines M times more permutations than the latter. Nevertheless, even though RFF considers a number of permutations that is orders of magnitude greater than those considered by PFF, it only explores the solution space around the initial permutation; PFF, on the other hand, examines permutations spread across the entire solution. Overall, these results indicate that PFF is effective in its use of computational resources in identifying solutions close to the lower bound.

V. CONCLUDING REMARKS

We have developed parameterized first-fit (PFF), a new heuristic for the SA problem that completely sidesteps the spectrum symmetry challenge and can identify near-optimal solutions to moderate-size networks in minutes. We plan to extend this work in two directions: we will explore the potential of PFF in larger-size networks, and we will combine PFF with the routing algorithms we developed in [22], [23] to tackle large RSA problems efficiently.

REFERENCES

[1] J. M. Simmons, *Optical Network Design and Planning*. Springer, 2008.

[2] J. Simmons and G. N. Rouskas, "Routing and wavelength (spectrum) allocation," in *B. Mukherjee, I. Tomkos, M. Tornatore, P. Winzer, and Y. Zhao (Editors), Springer Handbook of Optical Networks*, 2020.

[3] G. N. Rouskas, "Routing and wavelength assignment in optical WDM networks," in Wiley Encyclopedia of Telecommunications, 2001.

[4] B. Jaumard, C. Meyer, and B. Thiongane, "Comparison of ILP formulations for the RWA problem," *Optical Switching and Networking*, vol. 3-4, pp. 157–172, 2007.

[5] M. Klinkowski, P. Lechowicz, and K. Walkowiak, "Survey of resource allocation schemes and algorithms in spectrally-spatially flexible optical networking," *Optical Switc. and Netw.*, vol. 27, no. C, pp. 58–78, 2018.

[6] 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 Netw.*, vol. 13, pp. 34–48, July 2014.

[7] R. Dutta and G. N. Rouskas, "Traffic grooming in WDM networks: Past and future," *IEEE Network*, vol. 16, pp. 46–56, Nov/Dec 2002.

[8] R. Dutta and G. N. Rouskas, "A survey of virtual topology design algorithms for wavelength routed optical networks," *Optical Networks*, vol. 1, pp. 73–89, January 2000.

[9] H. Wang and G. N. Rouskas, "Hierarchical traffic grooming: A tutorial," *Computer Networks*, vol. 69, pp. 147–156, August 2014.

[10] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, pp. 16–23, November/December 2000.

[11] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas, "Spectrum assignment in optical networks: A multiprocessor scheduling perspective," *Journal of Optical Communications and Networking*, vol. 6, pp. 754–763, August 2014.

[12] R. Ramaswami and K. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 489–500, October 1995.

[13] B. Jaumard, C. Meyer, and B. Thiongane., "ILP formulations for the routing and wavelength assignment problem: Symmetric systems," in *Handbook of Optimization in Telecommunications*, pp. 637–677, Springer US, 2006.

[14] E. Yetginer, Z. Liu, and G. N. Rouskas, "Fast exact ILP decompositions for ring RWA," *Journal of Optical Communications and Networking*, vol. 3, pp. 577–586, July 2011.

[15] H. Zang, J. P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Optical Networks*, vol. 1, pp. 47–60, January 2000.

[16] Y. Zhu, G. N. Rouskas, and H. G. Perros, "A comparison of allocation policies in wavelength routing networks," *Photonic Network Communications*, vol. 2, pp. 265–293, August 2000.

[17] G. N. Rouskas and C. Bandikatla, "Recursive first fit: A highly parallel optimal solution to spectrum allocation.," *IEEE/Optica Journal of Optical Communications and Netw.*, vol. 14, pp. 165–176, April 2022.

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

[19] Z. Ortiz, G. N. Rouskas, and H. G. Perros, "Scheduling of multicast traffic in tunable-receiver WDM networks with non-negligible tuning latencies," in *Proceedings of SIGCOMM '97*, pp. 301–310, ACM, September 1997.

[20] V. Sivaraman and G. N. Rouskas, "HiPeR-ℓ: A High Performance Reservation protocol with ℓook-ahead for broadcast WDM networks," in *Proceedings of INFOCOM '97*, pp. 1272–1279, IEEE, April 1997.

[21] B. Chen, G. N. Rouskas, and R. Dutta, "Clustering methods for hierarchical traffic grooming in large-scale mesh wdm networks," *Journal of Optical Communications and Netw.*, vol. 2, pp. 502–514, August 2010.

[22] M. Fayez, I. Katib, G. N. Rouskas, T. F. Gharib, and H. Faheem, "A scalable solution to network design problems: Decomposition with exhaustive routing search," *Proc. IEEE GLOBECOM 2020*, Dec. 2020.

[23] G. N. Rouskas and C. Bandikatla, "Parameterized exhaustive routing with first fit for RSA problem variants," *Proc. IEEE GLOBECOM 2021*.