Chapter 4
Routing and Spectrum Allocation

Luis Velasco, Marc Ruiz, Kostas Christodoulopoulos, Manos Varvarigos, Mateusz Żotkiewicz, and Michal Pióro

Acronyms

B&B	Branch & Bound
BRKGA	Biased Random-Key Genetic Algorithm
CAPEX	Capital expenditures
CF	Central frequency
CG	Column generation
DAD	Dynamic Alternate Direction
DHL	Dynamic High expansion–Low contraction
DSP	Digital signal processing
GA	Genetic algorithm
GRANDE	Gradual Network Design
GRASP	Greedy Randomized Adaptive Search Procedure
ILP	Integer Linear Programming
LSO	Large-Scale Optimization
MILP	Mixed Integer Linear Programming
PLI	Physical layer impairments
PR	Path Relinking
QoT	Quality of the Transmission
RCL	Restricted Candidate List
RSA	Routing and spectrum allocation

L. Velasco (✉) • M. Ruiz
Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: lvelasco@ac.upc.edu

K. Christodoulopoulos • M. Varvarigos
University of Patras, Rio Achaia, Greece

M. Żotkiewicz • M. Pióro
Warsaw University of Technology, Warszawa, Poland

V. López, L. Velasco (eds.), *Elastic Optical Networks*, Optical Networks,
DOI 10.1007/978-3-319-30174-7_4

RWA	Routing and Wavelength Assignment
SEC	Spectrum Expansion/Contraction policy
WSON	Wavelength switched optical networks

To properly analyze, design, plan, and operate flex-grid networks, the routing and spectrum allocation (RSA) problem must be considered. The RSA problem involves two basic constraints: the continuity constraint to ensure that the allocated spectral resources are the same along the links in the route, and the contiguity constraint to guarantee that those resources are contiguous in the spectrum. Moreover, since advanced tunable transponders are envisioned for flex-grid networks, their configuration along with physical layer considerations need to be included in the optimization process. As a consequence of the RSA complexity, it is crucial that efficient methods are available for solving realistic problem instances in reasonable time. In this chapter, we review different variants of the RSA optimization problem; different methods to solve those variants are reviewed along with different requirements related to where those variants are applicable. Starting from a general RSA formulation, we analyze the network life-cycle and discuss different solving methods for the problems that arise at each particular network cycle: from offline to in-operation network planning. We tackle three representative use cases: (1) a use case for offline planning where a demand matrix need to be served taking into account physical layer impairments; (2) a use case for offline planning where a flex-grid network is designed and periodically upgraded; and (3) elastic bandwidth provisioning once the network is being operated.

4.1 Introduction

Before deploying a flex-grid network, certain activities need to be undertaken. Starting with inputs from the service layer and from the state of the already deployed resources, a planning phase needs to be carried out to produce recommendations that can be used to design the network for a given period of time. Next, the network design is verified and manually implemented. While the network is in operation, its capacity can be continuously monitored and the resulting data is used as input for the next planning cycle. In case of unexpected increase in demand or network changes, nonetheless, the planning process may be restarted.

Planning (i.e., network designing and dimensioning) generally consists in specifying the nodes and links to be installed and determining the equipment to be purchased to serve the foreseen traffic while minimizing network capital expenditures (CAPEX). To do so, traffic demands need to be routed over the network and a portion of the optical spectrum allocated to each of them, thus creating *lightpaths*. Note that the routing and spectrum allocation (RSA) problem is a part of the optimization problem that needs to be solved.

Once the network is being operated, provisioning algorithms need to solve the RSA in real time. In addition, Network reconfiguration requires solving optimization problems, where the obtained solutions are immediately implemented in the network [1]. Note that this is in contrast to traditional network planning, where results and recommendations require manual intervention and hence substantial time is needed until they are implemented in the network. *In-operation* planning involves the RSA problem; for example, recovery is a typical use case where the RSA problem encompasses a set of traffic demands.

Therefore, network planning problems can be solved *offline* since the network is not yet in operation and hence generally no strict time limit is applied to solve those problems. In contrast, when the network is *in operation*, stringent solving times are usually necessary.

Focusing on both, offline and in-operation planning, the contribution of this chapter is threefold: (1) we present two alternative Integer Linear Programming (ILP) formulations that can be used to model network planning variants of the RSA problem, named *link-path* and *node-link*. Several solving techniques to provide good trade-off between optimality and complexity are presented. In addition, a RSA algorithm for single demands is also reviewed; (2) two use cases for offline planning are presented: firstly the routing and spectrum allocation problem is extended with physical layer considerations, and secondly, the Gradual Network Design (GRANDE) problem, where the network is periodically updated to cope with yearly traffic increments, is presented; and (3) to illustrate online provisioning, the elastic bandwidth provisioning problem is presented.

4.2 Basic Offline Planning Problems

In this section we first introduce some basic concepts related to RSA and offline planning problems where the RSA need to be solved.

4.2.1 Basic Concepts

The RSA problem consists in finding a feasible route and spectrum allocation for a set of demands. Similarly to the Routing and Wavelength Assignment (RWA) problem in fixed-grid networks, the spectrum continuity constraint must be enforced. In the case of flex-grid, the spectrum allocation is represented by a *slot* and thus, in the absence of spectrum converters, the same slot must be used along the links of a given routing path. Besides, the allocated *slices* must be contiguous in the spectrum; this is called as spectrum contiguity constraint. The RSA problem was proved to be *NP*-complete in [2] and [3]. As a consequence, it is crucial that efficient methods are available to allow solving realistic problem instances in practical times.

Table 4.1 $C(d)$ precomputation

INPUT S, d	
OUTPUT $C(d)$	
1:	Initialize: $C(d) \leftarrow \mathbf{0}_{[|S|-nd+1 \text{ x } |S|]}$
2:	**for each** i in $[0, |S|-n_d]$ **do**
3:	**for each** s in $[i, i+n_d-1]$ **do**
4:	$C(d)[s]=1$
5:	**return** $C(d)$

Due to the spectrum contiguity constraint, RWA problem formulations developed for WDM networks are not applicable for RSA in flex-grid optical networks and they need to be adapted to include that constraint. Although several works can be found in the literature presenting ILP formulations for RSA, we rely on those in [4] since their approach, based on the assignment of slots, allows efficiently solving the RSA problem.

The definition of slots can be mathematically formulated as follows. Let us assume that a set of slots $C(d)$ is predefined for each demand d, which requests n_d slices. Let q_{cs} be a coincidence coefficient which is equal to 1 whenever slot $c \in C$ uses slice $s \in S$, and 0 otherwise. Hence, the spectrum contiguity constraint $\forall c \in C(d)$ is implicitly imposed by the proper definition of q_{cs} such that $\forall i, j \in S$: $q_{ci}=q_{cj}=1$, $i<j \Rightarrow q_{ck}=1, \forall k \in \{i, \ldots, j\}$, $\sum_{s \in S} q_{cs}=n_d$. We consider that each set $C(d)$ consists of all possible slots of the size requested by d that can be defined in S. Since $|C(n)|=|S|-(n-1)$, the size of the complete set of slots C that needs to be defined is $|C|=\sum_{n \in N} [|S|-n+1] < |N| \cdot |S|$.

The algorithm in Table 4.1 computes $C(d)$. Note that slot computation is trivial and thus no additional complexity is added to the precomputation phase.

Therefore, we can define the RSA problem as the problem that finds a proper lightpath, i.e., a route and a slot, for each demand from a given set so that the number of active slices in the assigned slot guarantees that the bitrate requested by each demand can be transported. Note that by precomputing the set of slots that can be assigned to each demand, the complexity added by the contiguity constraint is removed.

Finally, without loss of generality, we can consider that guard bands are included as a part of the requested spectrum, i.e., in n_d.

4.2.2 *Basic RSA Problem*

A basic RSA problem is to find a lightpath for every demand in a given traffic matrix with the objective of minimizing or maximizing some objective function. Several alternatives for this problem may exist, for instance, we can assume that the entire traffic specified by the traffic matrix needs to be served, or, alternatively, some demands can be blocked, in other words, not served. Additional characteristics, such as selecting the modulation format and/or accounting for the lightpath reach will be considered in the next subsection. The problem can be formally stated as follows.

Given:

- Connected graph $G(N, E)$, where N is the set of locations and E is the set of optical fibers connecting pairs of locations
- Characteristics of the optical spectrum (i.e., spectrum width and frequency slice width) and the set of modulation formats
- Traffic matrix D with the amount of bitrate exchanged between each pair of locations in N.

Output: route and spectrum allocation for each demand in D.

Objective: one or more among:

- Minimize the amount of bitrate blocked
- Minimize the total amount of used slices
- etc.

In the following, we present an ILP model for the above problem, based on the formulations in [4]. Note that since the topology is given, we can precompute a set of k distinct paths for each of the demands in the traffic matrix and hence, the formulation is usually known as *link-path* [5]. Moreover, because of the use of precomputed slots for each demand, we call this formulation Link-Path-Slot-Assignment (LP-SA).

The following sets and parameters are defined.Topology

N	Set of locations, index n
E	Set of fiber links, index e

Demands and Paths

D	Set of demands, index d. For each demand d, the tuple $\{o_d, t_d, b_d\}$ is given, where o_d and t_d are the origin and target nodes, and b_d is the bitrate in Gb/s
P	Set of precomputed paths, index p
$P(d)$	Subset of precomputed paths for demand d, $\lvert P(d)\rvert = k\ \forall d \in D$
r_{pe}	Equal to 1 if path p uses link e

Spectrum

S	Set of spectrum slices, index s
$C(d)$	Set of precomputed slots for demand d
q_{cs}	Equal to 1 if slot c uses slice s

The decision variables are:

w_d	Binary, equal to 1 if demand d cannot be served
x_{dpc}	Binary, equal to 1 if demand d is routed through path p and slot c

The LP-SA formulation is as follows:

$$(\text{LP}-\text{SA})\ \min \sum_{d\in D} b_d \cdot w_d \tag{4.1}$$

subject to:

$$\sum_{p\in P(d)} \sum_{c\in C(d)} x_{dpc} + w_d = 1, \quad \forall d \in D \tag{4.2}$$

$$\sum_{d\in D} \sum_{p\in P(d)} \sum_{c\in C(d)} r_{pe} \cdot q_{cs} \cdot x_{dpc} \le 1, \ \forall e \in E, s \in S \tag{4.3}$$

Objective function (4.1) minimizes the amount of bitrate that is not served (rejected). Constraint (4.2) ensures that a lightpath is selected for each demand provided that the demand is served; otherwise the demand cannot be served and therefore is rejected. Constraint (4.3) guarantees that every slice in every link is assigned to at most one demand.

The size of the LP-SA formulation is $O(|D|\cdot k\cdot |C|)$ variables and $O(|E|\cdot |S|+|D|)$ constraints. As an example, the size of the above formulation for the 22-node, 35-link BT network, considering $|S|=80$, $|D|=100$, and $k=10$, is 80,000 variables and 2900 constraints. Therefore, the size of this problem is noteworthy.

4.2.3 Topology Design as a RSA Problem

The previous RSA problems assumed that all links in E are installed but it is not guaranteed that they are sufficient to carry the full traffic demand, and if they are not we are minimizing the traffic that must be rejected. Another version of RSA is considered below; we assume that the links in E are sufficient to carry the demand specified by the traffic matrix and in fact not all of them are needed for that. Consequently, our objective is to minimize the number of links that are necessary to carry the entire demand. Since each installed link increases network CAPEX as a result of optical interfaces, including amplifiers, to be installed in the end nodes and some intermediate locations, minimizing the number of links in the resulting network topology would reduce CAPEX cost.

The problem can be formally stated as follows:

Given:

- A connected graph $G(N, E)$
- The characteristics of the optical spectrum and the set of modulation formats
- A traffic matrix D

Output:

- The route and spectrum allocation for each demand in D
- The links that need to be equipped

Objective: Minimize the number of links to be equipped to transport the given traffic matrix.

Note that we could precompute k distinct routes for each demand in the traffic matrix, as we did in the previous problem. However, since only part of the links will be eventually installed, the number of routes k to be precomputed for each demand would need to be highly increased to counteract the fact that some of the routes would become useless. For that very reason, we present an ILP formulation known as *node-link* [5] that includes routing computations within the optimization process. Similarly as before, because of the use of precomputed slots for each demand, we call this formulation as node-link slot-assignment (NL-SA).
A new parameter has been defined:

g_{ne}	Equal to 1 if link e is incident to node n

The decision variables are:

x_{dec}	Binary, equal to 1 if demand d uses slot c in link e
z_e	Binary, equal to 1 if link e is installed

The NL-SA formulation is as follows:

$$(\text{NL}-\text{SA}) \ \min \ \sum_{e \in E} z_e \tag{4.4}$$

subject to:

$$\sum_{e \in E} \sum_{c \in C(d)} g_{ne} \cdot x_{dec} = 1, \quad \forall d \in D, n \in \{o_d, t_d\} \tag{4.5}$$

$$\sum_{e \in E} \sum_{c \in C(d)} g_{ne} \cdot x_{dec} \leq 2, \ \ \forall d \in D, \ n \in N \setminus \{o_d, t_d\} \tag{4.6}$$

$$\sum_{\substack{e' \in E \\ e' \neq e}} g_{ne'} \cdot x_{de'c} \geq x_{dec}, \ \forall d \in D, c \in C(d), \ n \in N \setminus \{o_d, t_d\}, \ e \in E(n) \tag{4.7}$$

$$\sum_{d \in D} \sum_{c \in C(d)} q_{cs} \cdot x_{dec} \leq z_e, \ \forall e \in E, \ \ s \in S \tag{4.8}$$

Objective function (4.4) minimizes the amount of links to be installed. Constraints (4.5), (4.6), (4.7) specify the lightpath for every demand. The lighpath for demand $d \in D$ is specified by the links $e \in E$ and the channel $c \in C(d)$ for which $x_{dec} = 1$. Specifically, constraint (4.5) ensures that one lightpath for each demand is created with end nodes equal to the source and destination of demand. Constraint (4.6) guarantees that each lightpath is a connected set of links using the same slot along the route, whilst constraint (4.7) assures that the route does not contain any loop. Finally, constraint (4.8) prevents that any slice in any link is used by more than one demand, while installing the link when any slice is used.

The size of the NL-SA formulation is $O(|D| \cdot |E| \cdot |C|)$ variables and $O(|D| \cdot |C| \cdot |N| \cdot |E|)$ constraints. The size of this formulation for the BT network is 280,000 variables and 6,200,000 constraints, remarkably higher than in the LP-SA formulation.

4.2.4 Network Dimensioning as a RSA Problem

It is obvious that minimizing the number of links to be installed can be different than minimizing CAPEX, since some other costs need to be considered. For this very reason, the previous problem needs to be extended to take into account all the costs and to dimension all network resources.

The network dimensioning problem can be formally stated as follows.

Given:

- A connected graph $G(N, E)$
- The characteristics of the optical spectrum and the set of modulation formats
- A traffic matrix D
- The cost of every component, such as optical cross-connects (OXC), transponder (TP) types and regenerators specifying its capacity and reach. The cost of installing each link is also specified

Output:

- The route and spectrum allocation for each demand in D
- The links that need to be equipped
- Network dimensioning including the type of OXC, TPs, and regenerators in each location

Objective: Minimize the total CAPEX to transport the given traffic matrix.

Although we do not present any specific ILP model for this problem, it could be derived from the NL-SA formulation using an appropriate CAPEX model (see for example [6]). Notwithstanding, it is easy to realize that the size of the resulting formulation would be noticeable higher and might become intractable even using state-of-the-art computer hardware and the latest commercially available solvers, e.g., CPLEX [7]. In the next section we review some alternative solving techniques.

4.3 Solving Techniques

As we showed in the previous section, ILP or Mixed ILP (MILP) models for network planning might entail problems with literally thousands of millions of (integer or binary) variables. To deal with this complexity, in this section we present alternative methods to find near-optimal solutions.

Table 4.2 Path generation algorithm

```
INPUT: $G$, $D$
OUTPUT: *Solution*
1: $P^* \leftarrow$ initialize set of paths
2: $L \leftarrow$ initialize master problem (MILP with real variables)
3: while $P^* \neq \varnothing$ do
4:     Add new columns to $L$ from all paths in $P^*$
5:     $Solution \leftarrow$ Solve $L$
6:     $[\lambda, \pi] \leftarrow$ dual variables in $Solution$
7:     $P^* \leftarrow PricingProblem(G, D, [\lambda, \pi])$
8: $MILP \leftarrow L$ with binary variables
9: $Solution \leftarrow$ Solve $MILP$
```

4.3.1 Large-Scale Optimization

The objective of the Large-Scale Optimization (LSO) methods is to improve the exact methodology based on the classical Branch & Bound (B&B) algorithm for solving MILP formulations [8]. Among different methods, decomposition methods such as column generation (CG) and Benders decomposition have been successfully used for solving communications network design problems.

When solving large instances of problems based on precomputed variables (for example, LP-SA where paths are precomputed), it is necessary to include enough of those variables to ensure, at least, a near-optimal solution. Instead, CG provides a way to find a reduced set of variables producing high-quality solutions. Basically CG consists in two subproblems that are iteratively solved (see Table 4.2): the *restricted master* (or just master) *problem*, which is the linear relaxation of the original MILP that grows at each iteration with new variables, and the *pricing problem* that finds new variables to feed the restricted master. At each iteration, the pricing problem is solved taking as input data the dual variables of the restricted master. The iterative algorithm ends when no more variables are found, i.e., the current solution of the restricted master problem cannot be further improved. Since CG does not ensure integer optimal solutions, the B&B algorithm needs to be ultimately applied. In this regard, when CG is applied at each node of the B&B tree, the resulting algorithm is known as Branch & Price. Finally, note that in the context of networking problems, where variables are mainly paths, this technique is also known as path generation. This method has been recently proved to be an efficient way to generate precomputed lightpaths for link-path RSA formulations [9].

Benders decomposition method is an iterative procedure based on fixing a subset of *difficult* variables and solving the *master problem*, which includes the remaining variables. In order to reach the optimal solution for the original problem, a subproblem is solved for finding new constraints to be added to the master problem and improve the overall solution. Note that, in contrast to column generation, Benders decomposition adds inequalities to the linear formulation being solved, thus strengthening lower bounds and speeding up the convergence to integer optimal solutions. The combination of B&B with this and other methods to generate inequalities or cuts, such as cutting plane, derives into the Branch & Cut algorithm.

Nonetheless, when the time to find a solution is critical, which happens when the network is in operation, a better trade-off between solutions' quality and time-to-compute can be obtained by relaxing optimality condition to find near-optimal solutions much more quickly.

4.3.2 Metaheuristics

Heuristic algorithms are a practical way to produce suboptimal feasible solutions. In particular, metaheuristics (high-level strategies) guide a problem specific heuristic algorithm, to increase their performance avoiding the disadvantages of iterative improvement allowing escaping from local optima. Although a large variety of metaheuristic methods have appeared in the literature, we focus on describing two of them: the Greedy Randomized Adaptive Search Procedure (GRASP) [10] and the Biased Random-Key Genetic Algorithm (BRKGA) [11]. Some other *popular* metaheuristics are ant colony optimization, simulated annealing, and tabu search [12]. In addition, the Path Relinking (PR) intensification strategy is presented as a method to enhance heuristic solutions.

The GRASP procedure is an iterative two phase metaheuristic method based on a multi-start randomized search technique. In the first phase, a *constructive* algorithm is run to obtain a greedy randomized feasible solution of the problem. Roughly speaking, a solution is built by iteratively adding elements randomly chosen from a Restricted Candidate List (RCL) containing those elements with the best quality. The size of the RCL is determined by the parameter $\alpha \in [0, 1]$, being the extreme α values the pure greedy and the pure random configurations, respectively. Then, in the second phase, a local search technique to explore an appropriately defined neighborhood is applied in an attempt to improve the current solution. These two phases are repeated until a stopping criterion (e.g., number of iterations) is met, and once the procedure finishes, the best solution found over all GRASP iterations is returned. Table 4.3 presents an adaptation of the GRASP metaheuristic.

The BRKGA metaheuristic, a class of genetic algorithm (GA), has been recently proposed to effectively solve RSA-related optimization problems [13]. Compared to other metaheuristics, BRKGA has provided better solutions in shorter running times. As in GAs, each individual solution is represented by an array of *n genes* named *chromosome*, where each gene can take any value in the real interval $[0, 1]$. Each chromosome encodes a solution of the problem and a fitness value, i.e., the value of the objective function. A set of individuals, called a *population*, evolves over a number of *generations*. At each generation, individuals of the current genera-

Table 4.3 GRASP algorithm

```
INPUT $G$, $D$, $\alpha$, $maxIter$
OUTPUT $BestSol$
1:  $BestSol \leftarrow \emptyset$
2:  for $1..maxIter$ do
3:    $Sol \leftarrow \emptyset$; $Q \leftarrow D$
4:    while $Q \neq \emptyset$ do
5:      for each $d \in Q$ do
6:        evaluate the quality $q(d)$
7:      $q^{min} \leftarrow \min\{q(d) : d \in Q\}$; $q^{max} \leftarrow \max\{q(d) : d \in Q\}$
8:      $RCL \leftarrow \{d \in Q : q(d) \geq q^{max} - \alpha(q^{max} - q^{min})\}$
9:      Select an element $d$ from $RCL$ at random
10:     $Q \leftarrow Q \setminus \{d\}$; $Sol \leftarrow Sol \cup \{d\}$
11:   $Sol \leftarrow doLocalSearch(Sol)$
12:   $Sol$.fitness $\leftarrow$ computeFitness($Sol$)
13:   if $BestSol = \emptyset$ OR $Sol$.fitness > $BestSol$.fitness then
14:     $BestSol \leftarrow Sol$
15: return $BestSol$
```

tion are selected to mate and produce offspring, making up the next generation. In BRKGA, individuals of the population are classified into two sets: the *elite* set with those individuals with the best fitness values and *non-elite* set. Elite individuals are copied unchanged from one generation to the next, thus keeping track of good solutions. The majority of new individuals are generated by combining two elements, one elite and another non-elite, selected at random (crossover). An *inheritance probability* is defined as the probability that an offspring inherits the gene of its elite parent. Finally, to escape from local optima a small number of *mutant* individuals (randomly generated) are introduced at each generation to complete a population. A deterministic algorithm, named *decoder*, transforms any input chromosome into a feasible solution of the optimization problem and computes its fitness value. In the BRKGA framework, the only problem-dependent parts are the chromosome internal structure and the decoder, and thus, one only needs to define them to completely specify a BRKGA heuristic.

Metaheuristic methods can be extended to create hybrid methods that improve the performance of the original metaheuristic algorithm. One of the most extended hybrid methods consist in adding PR as an intensification strategy that explores trajectories connecting heuristic solutions. It starts at a so-called *initiating* solution and moves towards a so-called *guiding* solution. To ensure that PR is only applied among high-quality solutions, an elite set ES must be both maintained and cleverly managed during all iterations. With the attribute high-quality we are not only referring to their cost function value but also to the diversity they add to the set ES. GRASP+PR have been successfully used in many applications including flexgrid network defragmentation [14].

4.3.3 RSA Algorithm for Single Demands

Finally, let us analyze the special case where the RSA problem needs to be solved for a single demand. In this case, shortest paths algorithms (e.g., *k*-shortest paths [15]), can be adapted to include spectrum availability; in a second step, spectrum allocation can be realized using any heuristic algorithms (e.g., first fit, random selection, etc.).

In the *k*-shortest paths, each node *i* is labeled with the aggregated metric *m*(*i*) from the source node *o* and with its predecessor *pre*(*i*). Consequently, the route *o*-*i*, defined by the subset of links $E(o,i) \subseteq E$, can be computed repetitively visiting the predecessor node starting from *i*, until source node *o* is reached.

At this point, let η_{es} be equal to 1 if slice *s* in link *e* is free, 0 otherwise, and be *S*(*c*) the set of contiguous slices in slot *c*. Then, labels can be augmented with $\eta_s(i)$, the aggregated state of slice *s* ($\eta_s(i) = \prod_{e \in E(o,i)} \eta_{es}$) for each $s \in S$. The downstream node *j* of node *i* updates the label only if at least one slot is available as described in Eq. (4.9), i.e., only if $\sigma(i,j) = 1$.

$$\sigma\left(e=(i,j)\right)=\begin{cases}1 & \exists c \in C(d) : \eta_s(i)\cdot\eta_{es}=1 \quad \forall s \in S(c) \\ 0 & \text{otherwise}\end{cases} \tag{4.9}$$

Note that the complexity of the proposed spectrum availability extension is negligible. Besides, spectrum allocation is performed after the shortest route is found, adding flexibility to use any heuristic algorithms.

In the next sections we apply the presented techniques to different use cases for offline and in-operation planning.

4.4 Use Case I: Tunable Transponders and Physical Layer Considerations

In this use case we show how the ILP formulations in Sect. 4.2 can be extended to add extra considerations, such as physical layer impairments (PLIs).

Various implementations of flex-grid) transponders (referred to as bandwidth variable transponders—BVT) have been proposed [16], based on transmission techniques ranging from multi-carrier schemes, such as electrically or optically generated OFDM and Nyquist WDM, to single carrier schemes, usually employing some sort of digital signal processing (DSP) at the receiver but also at the transmitter side. These BVTs can adapt several transmission parameters, such as the modulation format, and/or the baudrate and/or the spectrum used. The term *software-defined optics* is also often used to describe these techniques.

The Quality of the Transmission (QoT) of a lightpath (measured by, for example, its BER) depends on its transmission parameters, the guardband left and the transmission parameters of the spectrum-adjacent lightpaths that share links with it. PLIs such as noise, dispersion, and interference effects accumulate and deteriorate the)QoT of the lightpath. To keep the QoT acceptable, a lengthy end-to-end connection may have to be regenerated and established in a multi-segment manner, with regenerators serving as “refueling stations” that restore signal quality. The multiple degrees of freedom present in flex-grid networks make connection establishment in such networks a more complicated problem than in fixed-grid WDM networks. The adaptability of the BVT and the interdependence between the transmission parameters, the PLIs and the transmission reach, add substantially to the complexity of the problem.

We propose a way to formulate the physical layer effects and the tunability of the transponders in a flex-grid network and outline an ILP algorithm that takes into account such input. We assume tunable transponders: each type of transponder has certain configurations, and each configuration has a specific optical reach, defined as the length at which the transponder can transmit with acceptable)QoT using that configuration. So, as also discussed above, the optical reach depends not only on the specific lightpath transmission configuration, but also on the presence of spectrum-adjacent interfering lightpaths, their transmission configurations and guardbands used. The number of combinations of possible configurations can be huge; also, PLI analytical models may not capture all effects or experimental measurements may be limited for some of the options. So, it seems that the only viable way to account for physical impairments is to resort to some sort of simplification that captures PLIs in a coarser but safe manner, reducing the parameters and the solution space without eliminating good solutions. In what follows we define the transmission reach assuming that a lightpath suffers worst case interference (four-wave-mixing, cross-phase modulation, cross-talk) by the adjacent lightpaths for a given transmission configuration and guardband distance.

Consider a BVT transponder of cost c that can be tuned to transmit r Gb/s using bandwidth of u spectrum slices and a guardband of g spectrum slots from its adjacent spectrum lightpaths, to reach l km distance with acceptable QoT. This defines a *physical feasibility function* $l = f_c(r, u)$ that captures PLIs and can be obtained experimentally or using analytical models [17, 18].

Using the functions f_c of the available transponders we define (*reach-rate-spectrum-cost*) transmission *tuples*, corresponding to feasible configurations of the transponders. The term “feasible” is used to signify that the tuple definition incorporates PLI limitations, while the cost parameter is used when there are transponders of different capabilities and costs. The above definition is very general and can be used to describe any type of flexible or even fixed-grid optical network. For example, the mixed-line-rate (MLR) fixed-grid network of [19] can be represented with the following transmission tuples: (3000 km, 10 Gb/s, 50 GHz, 1), (1600 km, 40 Gb/s, 50 GHz, 3), (800 km, 100 Gb/s, 50 GHz, 6). Using the above methodology the feasible transmission options can be enumerated so as to incorporate the physical layer effects.

The problem can be formally stated as follows:

Given:

- A connected graph $G(N, E)$
- The transmission configuration of the transponders, the physical layer and the availability or absence of regenerators, through the transmission tuples,

$t = (l_t, r_t, u_t, c_t)$ indicating a transmission of reach up to l_t, at rate r_t (Gb/s), using u_t spectrum slices, for a transponder of type (cost) c_t.
- A traffic matrix D

Output:

- The route and spectrum allocation for each demand in D

Objective: minimize the amount of bit-rate blocked

The formulation we present next is an extension of the one presented in [20] considering the definition of channel slots given in Sect. 4.2.1, and so it is called as transponder configuration-link-path-slot-assignment (TC-LP-SA). As in the previous LP-SA formulation, k paths are precalculated for each source-destination pair. For each path p and for each transmission tuple $t = (l_t, r_t, u_t, c_t)$, if the length of the path is shorter than l_t we define a feasible path-transmission tuple pair (p, t). We denote by T the set of all available tuples and by $T(p)$ the feasible tuples over path p.

Two key differences from the formulation presented in the previous section are: (1) in the new formulation different options for the number of spectrum slots that can be used to serve a demand are allowed and (2) demands can be broken into more than one lightpath, depending on the selected path and transmission tuple. The solution, in addition to assigning routes and slots to the demands, also selects the transmission configuration of the transponders.
The following sets and parameters have been defined:Topology

N	Set of locations, index n
E	Set of fiber links, index e

Demands and Paths

D	Set of demands, index d. For each demand d, the tuple $\{o_d, t_d, b_d\}$ is given, where o_d and t_d are the origin and target nodes, and b_d is the bitrate in Gb/s.
P	Set of precomputed paths, index p
$P(i, j)$	Subset of precomputed paths between nodes i and j, for all $(i, j) \in N^2$, with cardinality $\lvert P(i, j)\rvert = k$
$t = (l_t, r_t, u_t, g_t, c_t)$	A transmission tuple: l_t maximum reach, r_t rate (Gb/s), u_t spectrum slices, g_t guardband, transponder of type (cost) c_t. The total number of slices required is $n_t = u_t + g_t$
$T, T(p)$	Feasible transmission tuples for path p
(p, t)	A feasible path-transmission tuple.
r_{pe}	Equal to 1 if path p uses link e

Spectrum

S	Set of spectrum slices, index s
$C(t)$	Set of precomputed slots for transmission tuple t. Note that n_t is the size of the slot
q_{cs}	Equal to 1 if slot c uses slice s

The decision variables are:

w_d	Binary, equal to 1 if demand *d* cannot be served
x_{dptc}	Binary, equal to 1 if demand *d* uses path *p* and transmission tuple *t* and slot *c*

The TC-LP-SA formulation is as follows:

$$(\text{TC}\quad\text{LP}\quad\text{SA})\quad \min \sum_{d\in D} b_d \cdot w_d \tag{4.10}$$

subject to:

$$\sum_{n\in N}\sum_{p\in P(o_d,n)}\sum_{t\in T(p)}\sum_{c\in C(t)} r_t \cdot x_{dptc} + b_d \cdot w_d \geq b_d, \quad \forall d \in D \tag{4.11}$$

$$\sum_{n\in N}\sum_{p\in P(n,t_d)}\sum_{t\in T(p)}\sum_{c\in C(t)} r_t \cdot x_{dptc} + b_d \cdot w_d \geq b_d, \quad \forall d \in D \tag{4.12}$$

$$\sum_{i\in N}\sum_{p\in P(i,n)}\sum_{t\in T(p)}\sum_{c\in C(t)} x_{dptc} = \sum_{j\in N}\sum_{p\in P(n,j)}\sum_{t\in T(p)}\sum_{c\in C(t)} x_{dptc}, \quad \forall d \in D,\ n \in N \setminus \{o_d, t_d\},\ t \in T \tag{4.13}$$

$$\sum_{d\in D}\sum_{p\in P(d)}\sum_{t\in T(p)}\sum_{c\in C(t)} r_{pe} \cdot q_{cs} \cdot x_{dptc} \leq 1, \quad \forall e \in E,\ s \in S \tag{4.14}$$

The objective function (4.10), as in before, minimizes the amount of demands that cannot be served (rejected). Constraint (4.11) and (4.12) ensures that (one or more) lightpath(s) of enough rate are established to serve a demand provided that the demand is served; otherwise the demand cannot be served and therefore, is rejected. Constraint (4.13) conserves the lightpath flows at intermediate nodes where regenerators are placed. As written, it assumes that the regenerators would use the same transmission tuple as the initial transponder, and relaxes spectrum continuity, meaning that a different spectrum slots can be used after regeneration. Constraint (4.14) guarantees that every slice in every link is assigned to one demand at most.

The size of the TC-LP-SA formulation is $O(k \cdot |D|^2 \cdot |T| \cdot |C|)$ variables and $O(|E| \cdot |S| + |D| \cdot |N| \cdot |T|)$ constraints.

4.5 Use Case II: Gradual Network Design Problem

In this use case we focus on the planning phase)and study the problem of upgrading operators' core networks in order to extend its capacity as the traffic to be transported increases. We call this the Gradual Network Design (GRANDE) problem. In contrast to the models presented in section III, where the network planning was performed considering a green-field scenario, in the GRANDE problem the already deployed equipment can be reused to reduce upgrading CAPEX cost.

In the following subsections we first state the GRANDE problem and present its ILP formulation. As a result of its size, a method based on path generation is proposed. As an alternative, a BRKGA heuristic method is also presented. Illustrative numerical results are shown for two real network examples.

4.5.1 Problem Statement

The GRANDE problem can be formally)stated as follows.

Given:

- A connected graph $G(N, E)$, where N represents the set of potential OXC locations, and E is the set of links connecting pairs of locations
- Subsets $N_{in} \subseteq N$ and $E_{in} \subseteq E$ containing already installed OXCs and links, respectively
- The characteristics of the optical spectrum
- A traffic matrix D
- The cost of installing a new OXC and a new fiber link

Output:

- The network topology with the extra equipment needed to transport the set of demands D
- The route and spectrum allocation for each demand in D

Objective: Minimize the CAPEX cost resulting from upgrading the network to transport the given traffic matrix.

It is worth noting that to cope with new traffic requirements, manual operations causing service disruption are in general needed to implement the GRANDE solutions (see for example [21]). In this regard, the GRANDE problem focuses on optimizing traffic routing and network dimensioning admitting service disruption.

4.5.2 Mathematical Model

A node-link formulation was proposed in Sect. 4.2.3 for a network planning problem simpler than GRANDE. Although the use of node-link formulation may seem more effective than using link-path formulation when topology design is involved, it is hard to solve even for moderate-size network instances as a result its large size in terms of variables and constraints. Although link-path formulations need precomputed sets of paths for each demand, the path generation technique described in the next section can be applied to generate appropriate sets of paths that are required to achieve an optimal solution. For this very reason, we propose a link-path-based formulation for the GRANDE problem. The ILP model extends the LP-SA formulation presented in Sect. 4.2.2 adding constraints to tackle the already deployed equipment.

The approach we follow to minimize the upgrading CAPEX consists in setting the cost of the already installed equipment to zero and minimize total network CAPEX. CAPEX is divided into different components, such as the cost of installing a new node or a new link.

We note that the constraint to ensure that the whole traffic matrix is transported could result in the problem infeasibility when not enough resources are available. This constraint makes it difficult to apply path generation, so in the proposed formulation we allow demand rejection but with a large cost penalty. Note that when too few precomputed routes are used, the demands might be rejected.
In addition to the notation described so far, the following additional parameters are defined:

fn_n	Equal to 1 if an OXC is already installed in location n
fe_e	Equal to 1 if link e is already installed
cd_d	Penalty cost associated to demand d rejection
cn_n	Cost of installing a new OXC in location n
ct	Cost of adding a new link to an existing OXC
ce_e	Cost of installing link e

New decision variables are also defined:

y_n	Binary variable equal to 1 if an OXC is installed in location n

The ILP formulation for the GRANDE problem is as follows:

$$(GRANDE) \ \min \ \begin{aligned} &\sum_{d\in D} cd_d \cdot w_d + \sum_{e\in E}(1-fe_e)\cdot(ce_e + 2\cdot ct)\cdot z_e \\ &+\sum_{n\in N}(1-fn_n)\cdot cn_n \cdot y_n \end{aligned} \tag{4.15}$$

subject to:

$$\sum_{p\in P(d)}\sum_{c\in C(d)} x_{dpc} + w_d = 1, \quad \forall d\in D \tag{4.16}$$

$$\sum_{d\in D}\sum_{p\in P(d)}\sum_{c\in C(d)} r_{pe}\cdot q_{cs}\cdot x_{dpc} \le z_e, \quad \forall e\in E,\ s\in S \tag{4.17}$$

$$\sum_{d\in D}\sum_{p\in P(d)}\sum_{c\in C(d)} g_{ne}\cdot r_{pe}\cdot x_{dpc} \le |D|\cdot y_n, \quad \forall n\in N,\ e\in E \tag{4.18}$$

The objective function (4.15) minimizes the weighted sum of the cost of rejecting traffic (this term is assigned a coefficient c which is large with respect to the equipment costs) and the CAPEX cost of the additional equipment. It is worth highlighting that the value of the objective function is 0 when all demands can be routed using the already deployed nodes and links.

Table 4.4 Pricing problem algorithm

INPUT: G, D, S, dual variables [λ, π, μ]	
OUTPUT: P^*	
1:	**for** each d in D **do**
2:	$incCost \leftarrow 0$; $P^* \leftarrow \emptyset$
3:	**for** each slot c in $C(d)$ **do**
4:	Compute link metrics h_e from Eq. (4.20)
5:	$p^* \leftarrow$ Shortest path (o_d, t_d)
6:	Compute reduced cost u_{dp^*c} from Eq. (4.19)
7:	**if** $u_{dp^*c} > incCost$ **then**
8:	$incCost = u_{dp^*c}$
9:	$incPath = p^*$
10:	**if** $incPath \neq \emptyset$ **then**
11:	$P^* \leftarrow P^* \cup incPath$

Constraint (4.16) ensures that either a lightpath is assigned to each demand or the demand is rejected. Constraint (4.17) guarantees that each slice in each link is used to convey one lightpath at most and, additionally, installs those links conveying any demand. Constraint (4.18) assures that an OXC is installed in those locations that are used by at least one lightpath. Note that the number of lightpaths using a node cannot exceed the number of demands.

The size of the GRANDE model is $O(|D| \cdot k \cdot |C| + |N| + |E|)$ variables and $O(|E| \cdot |S| + |E| \cdot |N| + |D|)$ constraints, in line to that of the LP-SA formulation.

As discussed above, the presence of not installed nodes and links in the original topology makes it difficult to precompute path sets. On the one hand, paths crossing the installed resources do not contribute to the CAPEX increment but increase probability of rejecting demands. On the other hand, paths containing non-installed resources lead to solutions with lower or zero demand rejection but increase the CAPEX cost. A way to solve this issue is presented in the next section, where a path generation algorithm designed to find the best paths to minimize the objective function by simultaneous minimization of demand rejection penalties and by reducing network CAPEX.

4.5.3 Path Generation Algorithm

Aiming at finding the paths leading to good-quality solutions, we propose a path generation algorithm based on the one described in Table *4.4*. We have modified that algorithm to stop when either the pricing algorithm finds no more new paths or a maximum number of iterations (*maxIter*) is reached. This allows controlling the size of the problem, making this size proportional to the computational effort needed to obtain the optimal integer solution. Thus, it could be reasonable to stop generating paths and devote more time to solve the final primal problem.

The specific pricing problem behind path generation for GRANDE is as follows. First we derive the dual of the primal (master) problem (4.16)–(4.18). We define the following dual variables for the constraints of the relaxed GRANDE problem: λ_d unconstrained in sign for constraint (4.16), $\pi_{es} \geq 0$ for constraint (4.17), and $\mu_{ne} \geq 0$ for constraint (4.18). Finally, all variables in GRANDE are relaxed to be continuous $0 \leq x_{dpc}, w_d, y_n, z_e \leq 1$ in the master problem L, i.e., L contains the same set of variables and constraints as the original GRANDE formulation but with variables defined in the continuous domain.

From L, the dual problem can be easily derived from its Lagrangian function, which is obtained by moving the constraints to the objective function, multiplied by its associated dual variable. After grouping and re-ordering components, the Lagrangian function depending on primal and dual variables is as follows:

$$\begin{aligned} & L(x, y, z, \lambda, \pi, \mu, \gamma, \rho) \\ & = + \sum_{d \in D} w_d \cdot (cd_d - \lambda_d) + \sum_{n \in N} y_n \cdot \left((1 - fn_n) \cdot cn_n - |D| \cdot \sum_{e \in E} g_{ne} \cdot \mu_{ne} \right) \\ & \quad + \sum_{d \in D} w_d \cdot (cd_d - \lambda_d) + \sum_{n \in N} y_n \cdot \left((1 - fn_n) \cdot cn_n - |D| \cdot \sum_{e \in E} g_{ne} \cdot \mu_{ne} \right) \\ & \quad + \sum_{e \in E} z_e \cdot \left((1 - fe_e) \cdot (ce_e + 2 \cdot ct) - \sum_{s \in S} \pi_{es} \right) + \sum_{d \in D} \lambda_d \end{aligned} \tag{4.19}$$

Now the dual problem (D) is defined as a maximization problem in the dual space, where the variables in brackets are the primal variables related to each constraint in the dual.

$$(D) \quad \max \sum_{d \in D} \lambda_d \tag{4.20}$$

subject to:

$$\begin{aligned} [x_{dpc} \geq 0] \quad & \lambda_d \leq \sum_{e \in E} r_{pe} \cdot \left(\sum_{s \in S} q_{cs} \cdot \pi_{es} + \sum_{n \in N} g_{ne} \cdot \mu_{ne} \right), \\ & \forall d \in D,\ p \in P(d),\ c \in C(d) \end{aligned} \tag{4.21}$$

$$[y_n \geq 0] \quad |D| \cdot \sum_{e \in E} g_{ne} \cdot \mu_{ne} \leq (1 - fn_n) \cdot cn_n, \quad \forall n \in N \tag{4.22}$$

$$[z_e \geq 0] \quad \sum_{s \in S} \pi_{es} \leq (1 - fe_e) \cdot (ce_e + 2 \cdot ct) \quad \forall e \in E \tag{4.23}$$

Since the objective of the pricing problem is to add paths not considered so far, this turns into adding new dual constraints so as to the current optimal dual solution infeasible. Looking at the formulation of the dual problem, constraint (4.21) is the only one defined for the path variables; therefore, the sole condition that a candidate path p^* to be added to the problem must violate this constraint. Let u_{dp^*c} be the reduced cost of demand d using new path p^* and slot c; such p^* is a candidate path only if the reduced cost is strictly positive, i.e.,

$$u_{dp^*c} = \lambda_d - \sum_{e \in E} r_{pe} \cdot \left(\sum_{s \in S} q_{cs} \cdot \pi_{es} + \sum_{n \in N} g_{ne} \cdot \mu_{ne} \right) > 0 \tag{4.24}$$

Therefore, in light of Eq. (4.24), the pricing problem can be stated as follows: for each demand in D, find the path p^* and the slot $c \in C(d)$ hat maximize the reduced cost u_{dp^*c}, provided that $u_{dp^*c}>0$. Note that choosing the path with the highest positive reduced cost leads to the highest decrease rate in the objective function of the master problem.

An algorithm for solving the pricing problem is presented in Table *4.4*. For each demand d, a shortest path is computed for each of the slots in $C(d)$. For a given slot c, link metrics (h) used to compute shortest paths are setup as follows (line 4 in Table *4.4*):

$$h_e(c) = \sum_{s \in S} q_{cs} \cdot \pi_{es} + \sum_{n \in N} g_{ne} \cdot \mu_{ne} \tag{4.25}$$

Once the path is found (line 5), its reduced cost is computed using Eq. (4.24) and, if it is higher than the incumbent cost, the path is stored as incumbent path. After exploring all possible slots for the demand, the incumbent path is included in the set P^* containing the generated paths for all demands (lines 6–9).

We have presented a path generation method as an alternative to precomputing shortest paths following natural link metrics. However, when problem instances are excessively large preventing its application, alternative heuristic methods are needed. Next subsection presents the details of a BRKGA heuristic to solve the GRANDE problem.

4.5.4 BRKGA Heuristic

As pointed out in Sect. 4.3.2, the only problem-dependent parts of the BRKGA heuristic method are the chromosome structure and the decoder algorithm. Table *4.5* presents the pseudo-code of the decoder algorithm. Such a decoder uses chromosomes to sort the set of demands and hence each chromosome contains one gene for each demand d in D.

Starting from the topology with all nodes and links available, the metric of each node and link, which will be afterwards used by the RSA algorithm, are properly initialized to promote the use of the installed nodes and links (lines 1–3 in Table *4.5*). Next, the demands are sorted using the values of the genes in the input chromosome (line 4).

Table 4.5 Decoder algorithm

INPUT: G, D, Chromosome *chr*, *costInstall*	
OUTPUT: Solution, *fitness*	
1:	Initialize *Solution* with installed nodes and links
2:	Initialize metrics of nodes and links:
3:	if installed set to 0 otherwise to their cost
4:	Sort D according to genes in *chr*
5:	**for** each d in D **do**
6:	d.lightpath ← RSA(G, d)
7:	**if** d.lightpath = Ø **then**
8:	**return** *INFEASIBLE*
9:	allocate(G, d)
10:	*Solution.D* ← *Solution.D* ∪ d
11:	**if** *new nodes and/or link have been installed* **then**
12:	Set metrics of new installed equipment to 0
13:	*Solution.Equip* ← *Solution.Equip* ∪ {installed equip}
14:	*fitness* ← ComputeCAPEX (*Solution*)

Then, the decoder finds a lightpath for each of the demands following the given order (lines 5–6). Since the RSA algorithm finds a route with the minimum cost, the installed equipment will be reused before installing new nodes and/or links. After allocating the resources (line 9), metrics of the installed nodes and links are updated (lines 11–13). Finally, the fitness value is obtained by computing the CAPEX cost.

The next section evaluates performance of the methods proposed to solve GRANDE using network and traffic scenarios obtained from real network instances.

4.6 Use Case III: Elastic Bandwidth Provisioning

If we go back and consider the network life-cycle a planning-offline RSA algorithm (Sect. 4.2) is used to design the network. Then the network is gradually redesigned to adapt to traffic changes (Sect. 4.5). We envision a flexible network where dynamic traffic changes are accommodated at two different levels. The first level is the establishment of new connections that might need the deployment of new equipment (the GRANDE problem). Given the high rate that new generation transponders are expected to transmit (designs of 400 Gb/s or 1 Tb/s have appeared in the literature [16, 17]), relatively long periods of time will pass until a new connection requiring a new transponder is encountered. The second level is to absorb rate changes by adapting the tunable transponders, e.g., tuning the modulation format and/or the number of spectrum slots.

Therefore, in this use case we assume that lightpaths can expand/contract dynamically the spectrum they use to follow the small–medium scale dynamic traffic changes (e.g., daily traffic cycle). The slices that are freed by a lightpath can be

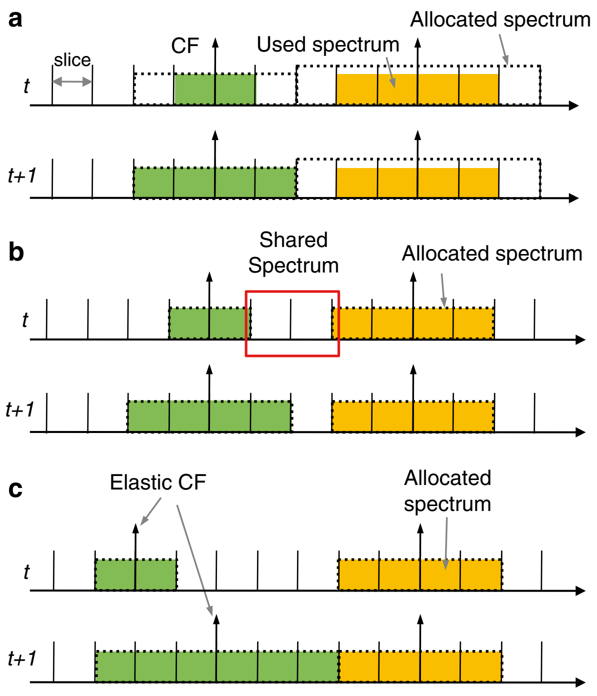

Fig. 4.1 Three spectrum allocation policies for time-varying traffic in a flex-grid network. Two time intervals are observed: *t* before and *t* + 1 after spectrum adaptation has been performed. (**a**) Fixed. (**b**) Semi-elastic. (**c**) Elastic

assigned to a different lightpath at different time instants, obtaining thus statistical multiplexing gains. Statistical multiplexing results in reducing the spectrum and the total number of slices required to serve a given traffic, or in the dual form of the problem, in reducing the blocking of the traffic changes for a given number of slices.

4.6.1 Spectrum Allocation Policies

Authors in [22] discerned three alternative spectrum allocation policies for time-varying traffic demands (reproduced in Fig. 4.1). The spectrum allocation policies put the following restrictions on the assigned central frequency (CF) and the allocated spectrum width, in particular:

- *Fixed* (Fig. 4.1a): both the assigned CF and spectrum width do not change in time. At each time period, demands may utilize either whole or only a fraction of the allocated spectrum to convey the bit-rate requested for that period.
- *Semielastic* (Fig. 4.1b): the assigned CF is fixed but the allocated spectrum may vary. Here, spectrum increments/decrements are achieved by allocating/releasing frequency slices symmetrically, i.e., at each end of the already allocated spectrum while keeping invariant the CF. The frequency slices can be

Table 4.6 Semielastic spectrum allocation algorithm

INPUT: $G(N,E), S, L, p, (s_p)^{req}$	
OUTPUT: $(s_p)'$	
1:	**if** $(s_p)^{req} \leq s_p$ **then**
2:	$(s_p)' \leftarrow (s_p)^{req}$
3:	**else**
4:	$L^+ \leftarrow \emptyset, L^- \leftarrow \emptyset$
5:	**for each** $e \in R_p$ **do**
6:	$L^- \leftarrow L^- \cup \{l \in L: e \in R_l$, adjacents$(l, p), f_l < f_p\}$
7:	$L^+ \leftarrow L^+ \cup \{l \in L: e \in R_l$, adjacents$(l, p), f_l > f_p\}$
8:	$s_{\max} \leftarrow 2*\min\{\min\{f_p - f_l - s_l, l \in L^-\}, \min\{f_l - f_p - s_l, l \in L^+\}\}$
9:	$(s_p)' \leftarrow \min\{s_{\max}, (s_p)^{req}\}$
10:	**return** $(s_p)'$

shared between neighboring demands, but used by, at most, one demand at a time interval.

- *Elastic* (Fig. 4.1c): asymmetric spectrum expansion/ reduction (with respect to the already allocated spectrum) is allowed and it can lead to short shifting of the central frequency. Still, the relative position of the lightpaths in the spectrum remains invariable, i.e., no reallocation in the spectrum is performed.

The problem of dynamic lightpath adaptation was addressed in [23] and it can be formally stated as:

Given:

- A core network topology represented by a graph $G(N, E)$, being N the set of optical nodes and E the set of bidirectional fiber links connecting two optical nodes; each link consists of two unidirectional optical fibers
- A set S of available slices of a given spectral width for every link in E
- A set L of lightpaths already established on the network; each lightpath l is defined by the tuple $\{R_l, f_l, s_l\}$, where the ordered set $R_l \subseteq E$ represents its physical route, f_l its central frequency and s_l the amount of frequency slices.
- A lightpath $p \in L$ for which spectrum adaptation request arrives and the required number of frequency slices, $(s_p)^{req}$.

Output: the new values for the spectrum allocation of the given lightpath p: $\{R_p, f_p, (s_p)'\}$ and $\{R_p, (f_p)', (s_p)'\}$, respectively, if the *Semielastic* and *Elastic* policy is used.

Objective: maximize the amount of bit-rate served.

For the *Fixed* spectrum allocation policy, the allocated spectrum does not change in time, therefore, any fraction of traffic that exceeds the capacity of the established lightpath is lost. Regarding the *Semielastic* and *Elastic* policies, the corresponding lightpath adaptation algorithms are presented in Tables 4.6 and 4.7, respectively. In the following, we discuss the details of these algorithms.

Table 4.7 Elastic spectrum allocation algorithm

```
INPUT: $G(N, E), S, L, p, (s_p)^{req}$
OUTPUT: $(f_p)', (s_p)'$
1:  if $(s_p)^{req} \leq s_p$ then
2:    $(s_p)' \leftarrow (s_p)^{req}$
3:    $(f_p)' \leftarrow f_p$
4:  else
5:    $L^+ \leftarrow \emptyset, L^- \leftarrow \emptyset$
6:    for each $e \in R_p$ do
7:      $L^- \leftarrow L^- \cup \{l \in L: e \in R_l, \text{adjacents}(l, p), f_l < f_p\}$
8:      $L^+ \leftarrow L^+ \cup \{l \in L: e \in R_l, \text{adjacents}(l, p), f_l > f_p\}$
9:    $s_{max} \leftarrow \min\{f_p - f_l - s_l, l \in L^-\} + \min\{f_l - f_p - s_l, l \in L^+\}$
10:   $(s_p)' \leftarrow \min\{s_{max}, (s_p)^{req}\}$
11:   $(f_p)' \leftarrow$ findSA_MinCFShifting $(p, (s_p)', L^+, L^-)$
12: return $(f_p)', (s_p)'$
```

- *Semielastic* algorithm: the elastic operation is requested for a lightpath p and the required amount of frequency slices to be allocated, maintaining f_p invariant, is given. Since flex-grid is implemented, $(s_p)^{\text{req}}$ must be an even number. If elastic spectrum reduction is requested, the tuple for lightpath p is changed to $\{R_p, f_p, (s_p)^{\text{req}}\}$ (lines 1–2). In the opposite, when an elastic expansion is requested, the set of spectrum-adjacent lightpaths at each of the spectrum sides is found by iterating on each of the links of the route of p (lines 4–7). The greatest value of available spectrum without CF shifting, s_{max}, is subsequently computed and the value of spectrum slices actually assigned to p, $(s_p)'$, is computed as the minimum between s_{max} and the requested one (lines 8–9). The tuple representing lightpath p is now $\{R_p, f_p, (s_p)'\}$.
- *Elastic* algorithm: here, the CF of p can be changed so the difference with the *Semielastic* algorithm explained above is related to that issue. Now, the value of s_{max} is only constrained by the amount of slices available between the closest spectrum-adjacent paths. Then, s_{max} is the sum of the minimum available slices along the links in the left side and the minimum available slices in the right side of the allocated spectrum (line 9). Finally, the returned value $(f_p)'$ is obtained by computing the new CF value so as to minimize CF shifting (line 11).

4.6.2 Spectrum Expansion/Contraction Policies

To enable the dynamic sharing of spectrum, we need to define Spectrum Expansion/Contraction (SEC) policies to regulate the way this is performed [24]. Under the proposed spectrum sharing framework, a connection shares the spectrum slots with its spectrum-adjacent connections.

- A first policy is the fixed policy, in which each connection is given a specific amount of spectrum and does not share it with other connections. This policy forms the baseline for the other policies that enable dynamic spectrum sharing among connections.
- A second policy is the Dynamic High expansion–Low contraction (DHL) policy, according to which a connection over path p wishing to increase its transmission rate first uses its higher spectrum slots until it reaches a slot already occupied by an upper spectrum-adjacent connection on some link of p. Then, if additional bandwidth is needed, it expands its lower slots until it reaches a slot that is occupied by some bottom spectrum-adjacent connection on some link of p. If the connection needs to increase further its rate and there is no higher or lower free slot space, blocking occurs (for the excess rate). Note that the DHL policy performs indirectly slot defragmentation, since it fills the free higher spectrum slots in every chance it gets. When a connection decreases its spectrum slots due to a reduction in its rate, we first release lower spectrum slots and, if these have been reduced to zero, we release higher slots.
- Another SEC policy is the Dynamic Alternate Direction (DAD) policy which aims at the symmetrical use of spectrum around the reference frequencies. With DAD, a connection wishing to increase its transmission rate alternates between using its higher and lower spectrum slots starting from its higher slots, until it reaches a slot already occupied by an upper or bottom, respectively, spectrum-adjacent connection. Then, if additional slots are needed, it expands towards the other direction, in which case the symmetry is lost. After that point, it always examines if it can expand towards the direction that uses fewer slots, using slots that were freed in the meantime by other connections. Blocking occurs if the connection needs more slots and there is no higher or lower free slot space. When a connection decreases its slots due to a reduction in its rate, we first release spectrum slots from the direction that has used more slots, and once we have an equal number of higher and lower slots, we decrease the lower spectrum slots. Thus, both expansion and contraction processes are designed to yield symmetrical spectrum utilization so that the CF of the connections does not frequently change and remains close to the reference frequency used when congestion is low.

More advanced SEC policies that consider the utilization of the spectrum-adjacent connections are proposed in [25].

Concluding Remarks

This chapter has reviewed RSA-related optimization problems and available solving techniques that can be used to solve those problems.

The notation used along the chapter was firstly introduced, where the spectrum contiguity constraint is modelled using slots, i.e., sets of contiguous frequency slices of fixed spectral width. Slots of the needed spectral width are assigned to optical connections. This greatly simplifies the spectrum allocation problem.

After the notation, the link-path, the transponder configuration-link path, and the node-link formulations were presented associated to simple offline planning problems. The sizes of these formulations were compared and we showed that those of the link-path formulations were much lower than those of the node-link. Nonetheless, the size of those really simple problems is so high that solving methods, apart from using commercial solvers for solving the mathematical models, need to be evaluated.

In view of the above, large-scale optimization techniques and metaheuristics were presented. Column generation and Bender's decomposition were presented to deal with large-scale problem instances, whereas GRASP and BRKGA metaheuristic frameworks to provide near-optimal solutions were detailed. In addition, an algorithm to compute the RSA problem for single-connection requests was introduced.

Three illustrative use cases were afterwards presented and solved: the provisioning problem considering PLIs, the GRANDE problem to design and periodically upgrade a flex-grid network, and the elastic bandwidth provisioning, where the spectrum allocated to a connection can be adapted to follow the time-varying required transmission rate.

References

1. L. Velasco, D. King, O. Gerstel, R. Casellas, A. Castro, V. López, In-operation network planning. IEEE Commun. Mag. **52**, 52–60 (2014)
2. K. Christodoulopoulos, I. Tomkos, E. Varvarigos, Elastic bandwidth allocation in flexible OFDM based optical networks. IEEE J. Lightwave Technol. **29**, 1354–1366 (2011)
3. Y. Wang, X. Cao, Y. Pan, A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks, in *Proceedings IEEE INFOCOM*, 2011
4. L. Velasco, M. Klinkowski, M. Ruiz, J. Comellas, Modeling the routing and spectrum allocation problem for flexgrid optical networks. Photon. Netw. Commun. **24**, 177–186 (2012)
5. M. Pióro, D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks* (Morgan Kaufmann, San Francisco, 2004)
6. O. Pedrola, A. Castro, L. Velasco, M. Ruiz, J.P. Fernández-Palacios, D. Careglio, CAPEX study for multilayer IP/MPLS over flexgrid optical network. IEEE/OSA J. Opt. Commun. Netw. **4**, 639–650 (2012)
7. CPLEX optimizer, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html
8. A. Land, A. Doig, An automatic method of solving discrete programming problems. Econometrica **28**, 497–520 (1960)
9. M. Ruiz, M. Pióro, M. Zotkiewicz, M. Klinkowski, L. Velasco, Column generation algorithm for RSA problems in flexgrid optical networks. Photon. Netw. Commun. **26**, 53–64 (2013)
10. T.A. Feo, M. Resende, Greedy randomized adaptive search procedures. J. Global Optim. **6**, 109–133 (1995)
11. J. Gonçalves, M. Resende, Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics **17**, 487–525 (2011)
12. M. Gendreau, J. Potvin, *Handbook of Metaheuristics*, 2nd edn. (Springer, Berlin, 2010)
13. L. Velasco, P. Wright, A. Lord, G. Junyent, Saving CAPEX by extending flexgrid-based core optical networks towards the edges (Invited Paper). IEEE/OSA J. Opt. Commun. Netw. **5**, A171–A183 (2013)
14. A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, J.P. Fernández-Palacios, D. Careglio, Dynamic routing and spectrum (re)allocation in future flexgrid optical networks. Comput. Netw. **56**, 2869–2883 (2012)

15. J. Yen, Finding the k shortest loopless paths in a network. Manag. Sci. **17**, 712–716 (1971)
16. A. Autenrieth, J.-P. Elbers, M. Eiselt, K. Grobe, B. Teipen, H. Griesser, Evaluation of technology options for software-defined transceivers in fixed WDM grid versus flexible WDM grid optical transport networks, in *ITG Symposium on Photonic Networks*, 2013
17. M. Svaluto Moreolo, J.J. Fabrega, L. Nadal, F.J. Vilchez, G. Junyent, Bandwidth variable transponders based on OFDM technology for elastic optical networks, in *International Conference on Transparent Optical Networks (ICTON)*, 2013
18. A. Klekamp et al., Limits of spectral efficiency and transmission reach of optical-OFDM superchannels for adaptive networks. IEEE Photon. Technol. Lett **23**(20), 1526–1528 (2011)
19. O. Rival, A. Morea, Cost-efficiency of mixed 10-40-100 Gb/s networks and elastic optical networks, in *OFC*, 2011
20. K. Christodoulopoulos, P. Soumplis, E. Varvarigos, Planning flexgrid optical networks under physical layer constraints. IEEE/OSA J. Opt. Commun. Netw. **5**, 1296–1312 (2013)
21. M. Ruiz, A. Lord, D. Fonseca, M. Pióro, R. Wessäly, L. Velasco, J.P. Fernández-Palacios, Planning fixed to flexgrid gradual migration: drivers and open issues. IEEE Commun. Mag. **52**, 70–76 (2014)
22. M. Klinkowski, M. Ruiz, L. Velasco, D. Careglio, V. Lopez, J. Comellas, Elastic spectrum allocation for time-varying traffic in flexgrid optical networks. IEEE J. Sel. Areas Commun. **31**, 26–38 (2013)
23. A. Asensio, M. Klinkowski, M. Ruiz, V. López, A. Castro, L. Velasco, J. Comellas, Impact of aggregation level on the performance of dynamic lightpath adaptation under time-varying traffic, in *Proc. IEEE International Conference on Optical Network Design and Modeling (ONDM)*, 2013
24. K. Christodoulopoulos, I. Tomkos, E. Varvarigos, Time-varying spectrum allocation policies in flexible optical networks. IEEE J. Sel. Areas Commun. **31**, 13–25 (2013)
25. E. Palkopoulou, I. Stakogiannakis, D. Klonidis, K. Christodoulopoulos, E. Varvarigos, O. Gerstel, I. Tomkos, Dynamic cooperative spectrum sharing in elastic networks, in *Optical Fiber Communications Conference (OFC)*, 2013