Noname manuscript No.
(will be inserted by the editor)

Integer Programming Models for the Routing and Spectrum Allocation problem*

Federico Bertero · Marcelo Bianchetti · Javier Marenco

Received: date / Accepted: date

Abstract One of the most promising solutions to deal with huge data traffic demands in large communication networks is given by flexible optical networking, in particular the *flexible grid* (flexgrid) technology specified in the ITU-T standard G.694.1. In this specification, the frequency spectrum of an optical fiber link is divided into narrow frequency *slots*. Any sequence of consecutive slots can be used as a simple channel, and such a channel can be switched in the network nodes to create a *lightpath*.

In this kind of networks, the problem of establishing lightpaths for a set of end-to-end demands that compete for spectrum resources is called the *routing and spectrum allocation problem* (RSA). Due to its relevance, this problem has been intensively studied in the last couple of years. It has been shown to be NP-hard [2, 9] and several models and formulations have been proposed, leading to different solution approaches.

In this work we explore integer programming models for RSA, analyzing their effectiveness over known instances. We resort to several modeling tech-

* Partially supported by D-TEC 0017/13 project.

F. Bertero
Instituto de Ciencias - Universidad Nacional de General Sarmiento
Juan María Gutiérrez 1150, 1613 Los Polvorines, Buenos Aires
Tel.: +5411 4469-7528
E-mail: fbertero@ungs.com

M. Bianchetti
Instituto de Ciencias - Universidad Nacional de General Sarmiento
Juan María Gutiérrez 1150, 1613 Los Polvorines, Buenos Aires
Tel.: +5411 4469-7528
E-mail: mbianchetti@dc.uba.ar

J. Marenco
Instituto de Ciencias - Universidad Nacional de General Sarmiento
Juan María Gutiérrez 1150, 1613 Los Polvorines, Buenos Aires
Tel.: +5411 4469-7528
E-mail: jmarenco@ungs.edu.ar

niques, in order to find natural formulations of this problem. Since integer programming techniques are known to provide successful practical approaches for several combinatorial optimization problems, the aim of this work is to explore a similar approach for RSA.

Keywords flexgrid optical networks · routing and spectrum allocation · integer programming · networks

Mathematics Subject Classification (2000) 90C10 · 94A05

1 Introduction

Optical networks represent a crucial infrastructure for our modern information society. Such networks use light as the communication medium between a sender node and a receiver node. For over two decades, the so-called *wavelength-division multiplexing* approach (WDM) has been the most popular technology in fiber-optic communications. WDM combines multiple wavelengths to simultaneously transport signals over a single optical fiber, and therefore enables capacity multiplication of the network. However, this technology must select the wavelengths from a fixed grid of frequencies specified by the International Telecommunication Union (ITU) and in general leads to an inefficient use of spectral resources.

In response to the sustained growth of data traffic volumes in communication networks, *flexgrid optical networks* have been introduced in the last few years, with the aim of enhancing the spectrum efficiency and thus enlarging network capacities. In flexgrid optical networks, the frequency spectrum of an optical fiber is divided into narrow frequency *slots* of fixed spectrum width. Any sequence of consecutive slots forms a *channel* that can be switched in the network nodes to create a *lightpath* (i.e., an optical connection). Flexgrid optical networks enable capacity gain by allocating minimum required bandwidth.

Following this approach, the elastic spectral resource can be represented using a number of contiguous frequency slots as shown in Figure 1.

The *routing and spectrum assignment problem* (RSA) [2, 3] consists in establishing the lightpaths for a set of end-to-end traffic demands that are expressed in terms of the number of required slots. Since lightpaths are determined by a route and a selected channel, RSA involves finding a route and assigning frequency slots to each demand. To comply with the ITU recommendation, the following constraints must to be respected in this setting:

- **slot continuity**: the slots assigned to a certain demand must remain the same on all the links of the corresponding route;
- **slot contiguity**: the slots allocated to each demand must be contiguous;
- **non-overlapping slot**: a slot can be assigned to various demands if the routes assigned to these demands do not share any link.

We now define RSA formally. Given a directed graph $G = (V, E)$ representing the optical fiber network, a set of demands $D = \{(s_i, t_i, v_i)\}_{i=1}^{k}$ –such

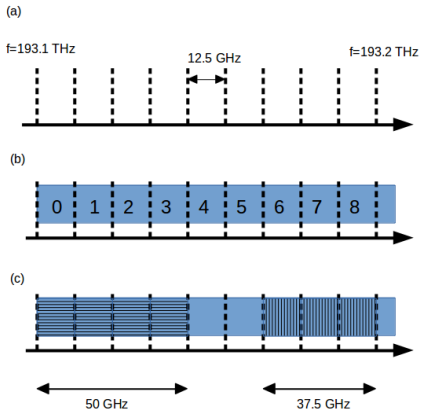

Fig. 1: (a), (b) Optical spectrum divided into frequency slots of fixed 12.5Ghz size. (c) Channels formed by sequences of consecutive slots

that each demand $i \in \{1, \ldots, k\}$ has a source $s_i \in V$, a target $t_i \in V$, and a volume $v_i \in \mathbb{Z}_+$– and a fixed number $S \in \mathbb{Z}_+$ of available slots, RSA consists in establishing a lightpath associated to each demand, in such a way that lightpaths do not overlap. In other words, each demand $i \in \{1, \ldots, k\}$ must be assigned a path $P_i \in E$ in G between s_i and t_i and an interval I_i consisting of v_i consecutive slots in $[1, S]$ in such a way that if $P_i \cap P_j \neq \emptyset$ then $I_i \cap I_j = \emptyset$, for any two demands $i \neq j$ (i.e., if the paths assigned to i and j share an arc, then the assigned slot intervals must be disjoint).

Figure 2 presents an example of the routing and spectrum allocation operation in a 4-node network. The instance asks for an assignment of a route and a sequence of contiguous slots for three demands given as follows: $d_1 = (v_1, v_3, 2)$, $d_2 = (v_2, v_4, 1)$ and $d_3 = (v_1, v_2, 3)$.

Many objective functions can be considered for RSA. It is common to try to minimize the highest used slot as well as the amount of used links. If the volume of the demands is not fixed by the constraints, then it is also natural to look for the minimum amount of used slots (and this is going to force to use just the volume of each demand). In this work we take as objective function the minimization of the total amount of edges used within all demands, namely the sum of the lengths of all the paths in the solution. This objective function avoids the generation of spurious paths in optimal solutions, since such a solution as few links as possible.

Since RSA is NP-hard, no polynomial-time algorithms for solving this problem to optimality are known, and it may be the case that no such algorithms exist (namely, if $P \neq NP$). In this case we can resort to heuristics but, if optimal solutions are needed, then exact algorithms may have impractical running times if the instances are not small-sized. In order to tackle problem sizes of real-world applications to optimality, algorithms have to be designed that rely on a deeper insight of the problem structure. Integer lin-

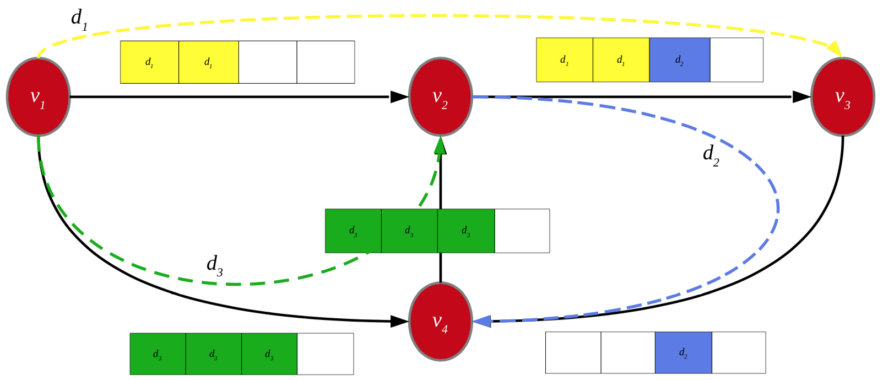

Fig. 2: Example of the routing and spectrum allocation operation for demands $d_1 = (v_1, v_3, 2)$, $d_2 = (v_2, v_4, 1)$ and $d_3 = (v_1, v_2, 3)$

ear programming (ILP) approaches, consisting of an in-depth investigation of polytopes associated with a combinatorial structure and the application of linear-programming-based cutting plane techniques, have been very successful in recent years for many combinatorial optimization problems. This motivates us to explore integer programming ideas for RSA.

The remainder of this paper is organized as follows. In Section 2 we provide the notation used throughout this work, then we show two ILP formulations taken from the literature that were used as a reference in the comparison presented in Section 3. We also present our ILP formulations for this problem, based on different modeling ideas that, to the best of our knowledge, have not been explored in previous works. In Section 3 we show performance evaluations of the presented ILP formulations. In order to do that, in the first part of the section we provide the computational results of testing the presented formulations over a big set of random generated instances. In the second part, we study the performance of our formulations and the ones from the literature over realistic instances. Finally, Section 4 closes the article with concluding remarks and lines for further development.

2 Integer programming formulations

In this section we present several ILP formulations for RSA. It must be noted that all formulations model the same variation of RSA, in spite of the fact that every model resorts to different sets of variables and constraints, hence has associated a different polytope. This is a typical practice in ILP (see, e.g., [1,5,6]): it is usually possible to provide more than one formulation for a given problem, and their practical performance cannot be predicted beforehand in a precise way. This leads the practitioner to state alternative ILP formulations for the same problem and to empirically evaluate running times in order to

evaluate them from a computational viewpoint. It is customary to employ the same objective function in all the considered models, in order to achieve a fair comparison.

As a consequence of such experiments, it is usual to choose the best-performing formulation, either in terms of the running time needed to achieve optimality or in terms of the optimality gap after a prespecified time limit. Even if it is not possible to achieve optimal solutions within practical running times, the formulations with the best optimality gaps provide a good starting point for the development of heuristics since the obtained feasible solutions have the best optimality guarantees. These considerations deem appropriate the development of alternative formulations and their empirical evaluation, which is the main focus of this work.

Figure 3 introduces the notation used throughout this work, including sets, constants, and variables.

Sets:

V	set of network nodes,
$E \subseteq V^2$	set of directed arcs (links),
D	set of demands,
$\delta^-(j) \subseteq E$	set of incoming arcs to node j,
$\delta^+(j) \subseteq E$	set of outgoing arcs from node j.
$E(j) \subseteq E$	set of arcs incidents to node j.

Constants:

$S \in \mathbb{Z}_+$	amount of available slots per arc,
$v(d) \in \{1, \ldots, S\}$	volume of demand d,
$s(d) \in V$	source of demand d,
$t(d) \in V$	destination of demand d.

Variables:

y_{de} Binary.	Equal to 1 if arc e is assigned to demand d, and 0 otherwise.
l_d Positive integer.	Contains the leftmost slot index assigned to demand d.
r_d Positive integer.	Contains the rightmost slot index assigned to demand d.
$p_{dd'}$ Binary.	Equal to 1 if for two demands d and d', $r_d < l_{d'}$, and 0 otherwise.
$n_{dd'}$ Binary.	Equal to 1 if the paths assigned to demands d and d' share an arc and $r_d < l_{d'}$, 0 otherwise.
r_{ds} Binary.	Equal to 1 if s is the first slot assigned to the demand d, and 0 otherwise.
l_{ds} Binary.	Equal to 1 if s is the last slot assigned to the demand d, and 0 otherwise.
x_{ds} Binary.	Equal to 1 if demand d uses slot s, and 0 otherwise.
u_{des} Binary.	Equal to 1 if demand d uses slot s over the link e, and 0 otherwise.
l_{des} Binary.	Equal to 1 if demand d uses the slot s over the link e, and s is the starting slot assigned to d, and 0 otherwise.
a_{ds} Binary.	Equal to 1 if demand d uses slots that are greater than s, and 0 otherwise.
b_{ds} Binary.	Equal to 1 if demand d uses slots that are smaller than s, and 0 otherwise.

Fig. 3: Notation table

2.1 Existing formulations

Several integer programming formulations for RSA can be found in the literature [2,4,7–9]. In most of them heuristics are presented and in order to reduce the complexity of ILP formulations a finite set of candidate paths is precomputed. In these last cases, to solve the problem to optimality precomputed sets must contain all allowable routes for each demand or column generation techniques must be applied.

Since we are interested in obtaining global optimal solutions with compact ILP formulations, we present two compact formulations taken from the stated literature that do not use predefined subsets of routes. We adapt these formulations in order to use the same objective function than the ILP formulations introduced in this work, so the computational comparison is fair.

Node-Link formulation involving slots (NLS)

This formulation is one of those presented in [8], adapted to the variables names we use in this work. The authors associate arcs and demands in order to define the paths, and the relation between demands and slots is modeled by partitioning the available slots for each demand into three disjoint groups: the non-used lower slots, the used slots and the non-used greater slots. In order to get these groups, for each demand $d \in D$ and each arc $e \in E$, the binary variable y_{de} is used in such a way that $y_{de} = 1$ if and only if the path associated to the demand d uses the arc e. Also, for each demand $d \in D$ they use the binary variables a_{ds}, x_{ds}, b_{ds}, in such a way that $a_{ds} = 1$ if and only if the demand d uses slots greater than s, $b_{ds} = 1$ if and only if the demand d uses slots lower than s and $x_{ds} = 1$ if and only if the demand d uses the slot s. In this setting, a feasible solution of RSA is an assignment of values to these variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} y_{de} \tag{1}$$

s.t.

$$\sum_{e \in \delta^-(s(d))} y_{de} - \sum_{e \in \delta^+(s(d))} y_{de} = 1 \qquad \forall d \in D \tag{2}$$

$$\sum_{e \in \delta^-(j)} y_{de} - \sum_{e \in \delta^+(j)} y_{de} = 0 \qquad \begin{array}{r} \forall d \in D \\ \forall v \in V \setminus \{s(d), t(d)\} \end{array} \tag{3}$$

$$y_{de} + x_{ds} + y_{d'e} + x_{d's} \leq 3 \qquad \begin{array}{r} \forall d, d' \in D,\ d \neq d', \\ \forall e \in E, \\ \forall s \in \{1, \ldots, S\} \end{array} \tag{4}$$

$$a_{ds} \geq a_{d,s+1} \qquad \forall d \in D,\ \forall s \in S \setminus \{S\} \tag{5}$$

$$b_{ds} \geq a_{d,s-1} \qquad \forall d \in D,\ \forall s \in S \setminus \{1\} \tag{6}$$

$$x_{ds} + a_{ds} + b_{ds} = 1 \qquad \forall d \in D,\ \forall s \in S \tag{7}$$

$$B \sum_{s \in S} x_{ds} \geq v(d) \qquad \forall d \in D \tag{8}$$

The objective function (1) aims to minimize the number of used links. The following two constraints, (2) and (3), express the classic law of conservation of flows. Constraints (4) assures that there not exists a slot shared by two demands that use the same link. While constraints (5) and (6) assure that channels are formed by consecutive slots, constraint (7) ensures the demand satisfaction.

Node-Link CA formulation (NL-CA)

This formulation is adapted from the one presented in [7], which is based on channel assignment removing the spectrum contiguity from RSA in order the reduce the problem complexity. In this formulation, the authors precompute all sets of candidate channels formed by consecutive slots. To find an optimal solution it is necessary to consider all possible channels for each demand (i.e. all channels of size $v(d)$ that can be defined in $\{1, \ldots, S\}$ for each demand d). For the sake of consistency, we present this model in terms of the variables used in this work, so we replace the $w-$variables employed in [7] by the $l-$variables considered in this work, without altering the model. Such an exchange in possible since assigning the channel $[s, \ldots, s + v(d) - 1]$ to the demand d is equivalent to asserting that s is the first slots assigned to d. About the objective function, we removed the $z-$variables -used to indicate opened links- and a family of constraints that set these variables to 1 when a link is used, so the comparison with the models presented in this work is fair. In this setting, a feasible solution of RSA is an assignment of values to these variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} \sum_{s=1}^{S-v(d)+1} l_{des} \tag{9}$$

s.t.

$$\sum_{e \in E(v)} \sum_{s=1}^{S-v(d)+1} l_{des} = 1 \qquad \forall d \in D,\ v \in \{s(d), t(d)\} \tag{10}$$

$$\sum_{e \in E(v)} \sum_{s=1}^{S-v(d)+1} l_{des} \leq 2 \qquad \forall d \in D,\ v \notin \{s(d), t(d)\} \tag{11}$$

$$\sum_{e' \in E(d):\ e' \neq e} l_{des} \geq l_{des} \qquad \forall d \in D,\ \forall s \in \{1, \ldots, S - v(d) + 1\},\ v \notin \{s(d), t(d)\},\ \forall e \in E(v) \tag{12}$$

The objective function (9) aims to minimize the number of used links. Constraints (10) - (12) compute the route through the optical topology and assign a slot to the demands. The constraint (10) assures that only one slot is used to transport the demand in any link incident to source and destination nodes. Finally constraints (11) and (12) perform the routing in intermediate nodes and implements the spectrum continuity constraint in intermediate nodes.

2.2 New formulations

RSA consists in establishing a route and a sequence of slots that respect the given constraints, for each demand. We consider two natural ways of modeling such a solution within an integer programming approach: either we represent the routing with a set of variables and the slot allocation with a second set of variables, or we represent both decisions with a single set of variables. We first state the models that use a set of variables for the assignment of demands to links and another for the assignment of demands to slots and then we present the models using a single set of variables for the assignment of demands to links and slots.

Inside each category, we also consider two ways of dealing with the *demand-channel* assignment: on the one hand we make an explicit *demand-slot*-assignment using variables that indicate if each slot is used by each demand. On the other hand, we make an implicit *demand-slot-range*-assignment using variables that indicate the first or/and last slot used by the demands. As the volume of the demands is fixed, if for example the slot s is the first slot used by the demand d, then the slots $s, \ldots, s + v(d) - 1$ will be also assigned to the demand d.

According to the above comments, we can group our ILP formulations into four families. On the one hand, the families resorting to different sets of variables for the routing and the slot allocation are classified to be either *Demand-Range* assignment formulations (DR) or *Demand-Slot* assignment formulations (DS), depending on the distinction of the preceding paragraph. On

the other hand, the families containing a single set of variables are classified to be either *Demand-Slot-Link* assignment formulations (DSL) or *Demand-Range-Link* assignment formulations (DRL), again depending on the distinction of the preceding paragraph. Given their variables, the existing formulations NLS and NL-CA can be classified into the families *Demand-Slot* (DS) and *Demand-Range-Link* (DRL), respectively.

In each of the following subsections we present one of the families above stated with a base ILP formulation associated and a set of modifications to it. Each of these modifications generates a new model that will be tested and analyzed in Section 3.

2.2.1 Demand-Range assignment formulations (DR)

Base formulation (DR-BF)

In this first model we associate arcs with demands in order to define the paths, and we also model the first and last slot allocated to each demand. To this end, for each demand $d \in D$ and each arc $e \in E$, we use the binary variable y_{de} in such a way that $y_{de} = 1$ if and only if the path associated to the demand d uses the arc e. Also, for each demand $d \in D$ we use the integer variables $l_d, r_d \in \{1, \ldots, S\}$, in such a way that the slot interval assigned to the demand d is given by $[l_d, r_d]$. Finally, in order to ensure that intervals associated with pairs of demands with non-disjoint paths do not overlap, we use the binary variable $p_{dd'}$ in such a way that for two demands d and d', if $r_d < l_{d'}$ then $p_{dd'} = 1$. In this setting, a feasible solution of RSA is an assignment of values to these variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} y_{de} \tag{13}$$

s.t.

$$\sum_{e \in \delta^-(j)} y_{de} - \sum_{e \in \delta^+(j)} y_{de} = \begin{cases} \text{-1} & \text{if } j = s(d) \\ 1 & \text{if } j = t(d) \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in V, \forall d \in D \tag{14}$$

$$p_{dd'} + p_{d'd} = 1 \qquad \forall d, d' \in D,\ d \neq d' \tag{15}$$

$$r_d + 1 \leq l_{d'} + S(3 - p_{dd'} - y_{de} - y_{d'e}) \qquad \forall d, d' \in D,\ d \neq d',\ \forall e \in E \tag{16}$$

$$r_d - l_d + 1 = v(d) \qquad \forall d \in D \tag{17}$$

$$1 \leq l_d \leq r_d \leq S \qquad \forall d \in D \tag{18}$$

With the objective function (13) we aim to minimize the total number of arcs in all assigned paths.

Constraints (14) ensure that the y-variables define a path from the source to the destination of each demand, by imposing flow conservation constraints to each node in the network. Note that (14) do not prevent the formation of

spurious cycles, but we do not include constraints explicitly forbidding this situation since the formation of cycles does not affect feasibility and the addition of (exponentially many) cycle-breaking constraints may affect solution times.

Constraints (15) and (16) ensure that if the demands d and d' share one arc, then either $r_d < l_{d'}$ or $r_{d'} < l_d$. Indeed, if d and d' share the edge e, then (15) implies that either $p_{dd'} + y_{de} + y_{d'e} = 3$ or $p_{d'd} + y_{de} + y_{d'e} = 3$, thus imposing the desired condition. Finally, constraints (17) and (18) ensure that the number of slots assigned to each demand satisfies its required volume.

One slot bound (DR-OSB)

Constraints (17) allow us to eliminate the r-variables, since these variables can be written in terms of the l-variables. Then, a re-statement of the previous formulation can be obtained by replacing the constraints (17) and (16) with the following ones.

$$l_{d'} \geq l_d + v(d) - S(3 - p_{dd'} - y_{de} - y_{d'e}) \qquad \forall d, d' \in D, \forall e \in E \tag{19}$$

Alternative order variable (DR-AOV)

It could be useful to replace the $p-$variables with ones that help to ensure that intervals associated with pairs of demands with non-disjoint paths do not overlap. The variable $n_{dd'}$ equals 1 if the paths assigned to d and d' share an arc and $r_d < l_{d'}$. By assigning values to the variables satisfying (13), (14), (17), (18), and the following constraints, we obtain a feasible solution to RSA.

$$n_{dd'} + n_{d'd} \geq y_{de} + y_{d'e} - 1 \qquad \forall d, d' \in D,\ d \neq d',\ \forall e \in E \tag{20}$$

$$r_d + 1 \leq l_{d'} + S(1 - n_{dd'}) \qquad \forall d, d' \in D,\ d \neq d' \tag{21}$$

Constraints (20) specify that if two demands d and d' share a link (i.e., there exists some e with $y_{de} = y_{d'e} = 1$) then $n_{dd'} = 1$ or $n_{d'd} = 1$. Constraints (21) impose that if $n_{dd'} = 1$ then $r_d + 1 \leq l_{d'}$, therefore the slots assigned to d' have larger indices than the slots assigned to d, and if $n_{d'd} = 1$ then the opposite happens. As both events are disjoint, we cannot have $n_{dd'} = 1$ and $n_{d'd} = 1$ simultaneously. Thus we guarantee that any pair of two different demands sharing a link are not assigned to the same slot.

Binary slot assignment (DR-BSA)

An alternative to the assignment of slots to demands through the integer $l-$ and $r-$ variables is the use of binary variables. We implement this assignment by using the binary variable l_{ds} for each demand $d \in D$ and each slot $s \in$

$\{1, \ldots, S - v(d) + 1\}$, in such a way that $l_{ds} = 1$ if s is the first slot assigned to the demand d, and $l_{ds} = 0$ otherwise. If $l_{ds} = 1$, then the slots $s, \ldots, s+v(d)-1$ are assigned to the demand d. The use of these variables allows us to state the following formulation that uses constraints (14) and defines new constraints:

$$\sum_{s=1}^{S-v(d)+1} l_{ds} = 1 \qquad \forall d \in D \tag{22}$$

$$\sum_{s'=s}^{s+v(d)-1} l_{d's'} \leq 3 - y_{de} - y_{d'e} - l_{ds} \qquad \begin{array}{l} \forall d, d' \in D,\ d \neq d', \forall e \in E, \\ \forall s \in \{1, \ldots, S - v(d) + 1\} \end{array} \tag{23}$$

Constraints (22) ensure that each demand has exactly one *starting slot*. To deal with the non-overlapping restriction, constraints (23) guarantee that if s is the starting slot assigned to the demand d, then the next $v(d) - 1$ slots on the links used by d are assigned to d too (i.e., none of them is the starting slot of any other demand sharing a link with d).

Stronger constraints (DR-SC)

Constraints (23) ensure in a single expression that the slots assigned to the demand d are consecutive. We can formulate the same assertion by replacing the summation in (23) by $v(d)$ new inequalities, as follows.

$$l_{d'(s+i)} \leq 3 - y_{de} - y_{d'e} - l_{ds} \qquad \begin{array}{r} \forall d, d' \in D,\ d \neq d',\ \forall e \in E, \\ \forall s \in \{1, \ldots, S - v(d) + 1\}, \\ \forall i \in \{0, \ldots, v(d) - 1\} \end{array} \tag{24}$$

Constraints (24) impose that if $y_{de} + y_{d'e} + l_{ds} = 3$ then $l_{d's'} = 0$ for $s' = s, \ldots, s+v(d)-1$. In other words, if two different demands d and d' share a link e, and s is the first slot used by d, then no slot in $\{s, \ldots, s + v(d) - 1\}$ can be the first slot of d'.

2.2.2 Demand-Slot assignment formulations (DS)

Base formulation (DS-BF)

This is the first of a second family of models explicitly representing each slot assigned to every demand, instead of assigning slot ranges with the $l-$ and $r-$ variables. For each demand $d \in D$ and each slot $s \in \{1, \ldots, S\}$, the binary variable x_{ds} is equal to 1 if the demand d uses the slot s, and $x_{ds} = 0$ otherwise. To associate arcs with demands, we use the y-variables as in the previous models. In this setting, a feasible solution to RSA will be an assignment of values to these two families of variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} y_{de} \tag{25}$$

s.t.

(14)

$$y_{de} + x_{ds} + y_{d'e} + x_{d's} \leq 3 \qquad \begin{array}{r} \forall d, d' \in D,\ d \neq d', \\ \forall e \in E, \\ \forall s \in \{1, \ldots, S\} \end{array} \tag{26}$$

$$x_{d,S+1} = 0 \qquad \forall d \in D \tag{27}$$

$$\sum_{s'=1+s-v(d)}^{s} x_{ds'} \geq v(d)(x_{ds} - x_{d(s+1)}) \qquad \begin{array}{r} \forall d \in D, \\ \forall s \in \{v(d), \ldots, S\} \end{array} \tag{28}$$

$$\sum_{s=1}^{S} x_{ds} = v(d) \qquad \forall d \in D \tag{29}$$

Constraints (14) are the same used in the previous models to define a path from the source to the destination of each demand. To ensure that if two demands share a link then they cannot share a slot, we need at least one of the variables y_{de}, x_{ds}, $y_{d'e}$ and $x_{d's}$ not to be activated, and constraints (26) enforce this condition. Constraints (27) and (28) guarantee slots contiguity. Constraints (28) assert that if the slot s is assigned to the demand d and the neighboring slot $s+1$ is not assigned to the demand d, then the $v(d)-1$ slots to the left of s must be assigned to d too. In order to simplify the notation, we assume the existence of the variable $x_{d,S+1}$ taking value 0. Finally, in order to satisfy the volume required for each demand, we include constraints (29), which force the amounts of slots of a demand to be at least the required volume.

Alternative contiguity constraints (DS-ACC)

Based on the fact that constraints (28) can be unfolded we replace these constraints with the following ones:

$$x_{ds} + x_{ds'} \ \leq \ x_{d,s+1} + 1 \qquad \begin{array}{r} \forall s, s' \in \{1, \ldots, S\}, \\ s' > s,\ \forall d \in D \end{array} \tag{30}$$

Constraints (30) impose that if $x_{ds} - x_{d,s+1} = 1$ then s must be the last slot assigned to the demand d, hence $x_{ds'} = 0$ for $s' > s$. This ensures that all slots assigned to a demand are consecutive.

Stronger contiguity constraints (DS-SCC)

These new constraints are based on another way to state constraints (28).

$$\sum_{\substack{s\in\{1,\dots,S\}:\\ s\ \equiv\ j\ (mod\ v(d))}} x_{ds} \ = \ 1 \qquad \begin{array}{r}\forall d \in D,\\ \forall j \in \{1,\dots,v(d)\}\end{array} \tag{31}$$

$$\sum_{\substack{s\in\{1,\dots,i\}:\\ s\ \equiv\ i\ (mod\ v(d))}} x_{ds} \ \geq \sum_{\substack{s\in\{1,\dots,i-1\}:\\ s+1\ \equiv\ i\ (mod\ v(d))}} x_{ds} \qquad \begin{array}{r}\forall d \in D,\\ S' = S + 1 - v(d),\\ \forall i \in \{1,\dots,S'\}\end{array} \tag{32}$$

Constraints (31) and (32) ensure that for each demand and for each link there are exactly $v(d)$ consecutive variables taking value 1, and that the remaining variables take null values. This fact is not straightforward, therefore in the Appendix we provide a proof showing that these constraints suffice to enforce such a configuration.

2.2.3 Demand-Slot-Link formulations (DSL)

Base formulation (DSL-BF)

As explained before, it is possible to model the routing and the resource allocation with just one family of variables. The u-variables relate demands with links and slots in the following way. For every demand $d \in D$, every link $e \in E$ and every slot $s \in \{1,\dots,S\}$, we use the variable u_{des} such that $u_{des} = 1$ if the demand d uses the slot s over the link e. In this setting, a feasible solution to RSA is an assignment of values to these variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} \sum_{s=1}^{S} u_{des}/v(d) \tag{33}$$

s.t.

$$\sum_{e \in \delta^-(j)} u_{des} - \sum_{e \in \delta^+(j)} u_{des} = 0 \qquad \begin{array}{r} \forall j \in V,\ j \neq s(d),\ j \neq t(d), \\ \forall d \in D,\ \forall s \in \{1, \ldots, S\} \end{array} \tag{34}$$

$$\sum_{e \in \delta^-(j)} u_{des} = 0 \qquad \begin{array}{r} \forall j \in V,\ j = s(d), \\ \forall d \in D,\ \forall s \in \{1, \ldots, S\} \end{array} \tag{35}$$

$$\sum_{e \in \delta^+(j)} \sum_{s=1}^{S} u_{des} \geq v(d) \qquad \begin{array}{r} \forall j \in V,\ j = s(d), \\ \forall d \in D \end{array} \tag{36}$$

$$\sum_{d \in D} u_{des} \leq 1 \qquad \forall e \in E,\ \forall s \in \{1, \ldots, S\} \tag{37}$$

$$u_{de,S+1} = 0 \qquad \forall d \in D,\ \forall e \in E \tag{38}$$

$$\sum_{s'=s-v(d)+1}^{s} u_{des'} \geq v(d)(u_{des} - u_{de,s+1}) \qquad \begin{array}{r} \forall d \in D,\ \forall e \in E, \\ \forall s \in \{v(d), \ldots, S\} \end{array} \tag{39}$$

The objective function (33) aims to minimize the total number of links used in the routing.

Constraints (34), (35), and (36) ensure that the u-variables define a path from the source to the destination of each demand. Specifically, (34) impose flow conservation constraints to each node in the network, except for the source and sink of the corresponding demand. Constraints (35) prevent using arcs entering the source node of the corresponding demand, and constraints (36) avoid empty flows (i.e., all the u-variables taking null values).

The main advantage of the u-variables is that it is easy to guarantee that each slot is used at most once in every link. Indeed, constraints (38) and (39) ensure slot contiguity and demand satisfaction in a quite natural way. When $u_{des} - u_{de,s+1} \neq 0$ we have that the slot s is the last slot used by the demand d on the link e, so we ensure that all the variables u_{dei} for i in the interval between $s - v(d)$ and s are activated. Because we need to cover every possible combination of two consecutive slots, including the rightmost one, we include in this model the fictional variable $u_{de,S+1}$, which always takes value zero.

Alternative satisfiability and contiguity constraints (DSL-ASCC)

Constraints (38) and (39), which ensure slot contiguity and satisfiability of demands, can be replaced by the following alternative constraints

$$v(d)\ u_{des} \leq \sum_{s'=1}^{S} u_{des'} \qquad \forall d \in D,\ \forall e \in E,\ \forall s \in \{1,\ldots,S\} \tag{40}$$

$$u_{des_1} + u_{des_2} \leq u_{de(s_1+1)} + 1 \qquad \begin{array}{r}\forall d \in D,\ \forall e \in E,\\ \forall s_1, s_2 \in \{1,\ldots,S\},\ s_2 > s_1\end{array} \tag{41}$$

If the demand d uses the slot s on the link e, then $u_{des} = 1$ and constraint (40) guarantees that the number of slots used for d in e is greater than or equal to $v(d)$. Therefore, these constraints ensure that each demand is satisfied. Moreover, if the demand d is assigned the slots s_1 and s_2 (with $s_1 < s_2$) on the link e, then every slot in the range $\{s_1,\ldots,s_2\}$ must also be assigned to d on the link e, and this is enforced by constraints (41).

Adding slots bounds (DSL-ASB)

Instead of using the integer variables l_d and r_d to represent the slot interval assigned to the demand d, in this model we resort to binary variables associated with each demand and each slot. For $d \in D$ and $s \in \{1,\ldots,S\}$, we use the binary variable l_{ds} (resp. r_{ds}) in such a way that $l_{ds} = 1$ (resp. $r_{ds} = 1$) if s is the first (resp. last) slot assigned to d. In this setting, formulation DSL-ASB uses constraints (33), (34), (35), (36), (37), (40) and the following constraints:

$$\sum_{s=1}^{S} s(r_{ds} - l_{ds}) = v(d) \qquad \forall d \in D \tag{42}$$

$$\sum_{s=1}^{S} l_{ds} = 1 \qquad \forall d \in D \tag{43}$$

$$\sum_{s=1}^{S} r_{ds} = 1 \qquad \forall d \in D \tag{44}$$

$$u_{des} \leq \sum_{s'=1}^{s} l_{ds'} \qquad \forall d \in D,\ \forall e \in E,\ \forall s \in \{1,\ldots,S\} \tag{45}$$

$$u_{des} \leq \sum_{s'=s}^{S} r_{ds'} \qquad \forall d \in D,\ \forall e \in E,\ \forall s \in \{1,\ldots,S\} \tag{46}$$

If s_1, s_2 are the leftmost and rightmost slots assigned to the demand d, then $l_{ds_1} = r_{ds_2} = 1$ while $l_{ds'} = 0$ for all slots s' different from s_1, and $f_{ds'} = 0$ for all slots s' different from s_2. Then the left-hand-side of constraints (42) reduces to $s_2 - s_1$, implying that the range assigned to the demand satisfies its volume. Constraints (43) and (44) guarantee that each demand has exactly one leftmost and one rightmost slot. Finally, constraints (45) and (46) relate the u-variables with the l- and r- variables, by assuring that any slot used by a demand belongs to the range specified by the corresponding l- and r-variables.

2.2.4 Demand-Range-Link formulations (DRL)

Base formulation (DRL-BF)

Instead of explicitly assigning slots to demands and links, in this formulation we represent the starting slot assigned to each demand in each link. To this end, we use the binary variable l_{des} for each $d \in D$, $e \in E$, and $s \in \{1, \ldots, S - v(d) + 1\}$, in such a way that $l_{des} = 1$ if and only if the demand d uses the slot s over the link e, and s is the starting slot assigned to d. In this setting, a feasible solution to RSA is an assignment of values to the l-variables satisfying the following constraints.

$$\min \sum_{d \in D} \sum_{e \in E} \sum_{s=1}^{S} l_{des} \tag{47}$$

s.t.

$$\sum_{e \in \delta^-(j)} l_{des} - \sum_{e \in \delta^+(j)} l_{des} = 0 \qquad \begin{array}{r} \forall j \in V,\, j \neq s(d),\, j \neq t(d), \\ \forall d \in D,\, \forall s \in \{1, \ldots, S\} \end{array} \tag{48}$$

$$\sum_{e \in \delta^-(j)} \sum_{s=1}^{S} l_{des} = 0 \qquad \forall j \in V,\, j = s(d),\, \forall d \in D \tag{49}$$

$$\sum_{e \in \delta^+(j)} \sum_{s=1}^{S} l_{des} = 1 \qquad \forall j \in V,\, j = s(d),\, \forall d \in D \tag{50}$$

$$\sum_{d' \in D : d \neq d'} \sum_{s'=s}^{\min\{S,\, s+v(d)-1\}} l_{d'es'} \leq S(1 - l_{des}) \qquad \begin{array}{r} \forall d \in D,\, \forall e \in E, \\ \forall s \in \{1, \ldots, S\} \end{array} \tag{51}$$

In this formulation we try to minimize the objective function (47), looking to minimize the total number of links used in the routing.

Constraints (48), (49), and (50) ensure the existence of a feasible path for each demand, eliminate cycles starting and ending in the source node, and guarantee that each demand is assigned to exactly one starting slot, respectively.

Constraints (51) ensure that if the demand d is assigned to the slot s as its starting slot over the link e, then s and the next $v(d) - 1$ slots cannot be assigned to any other demand d' over the same link, thus avoiding slot overlapping.

2.3 Discussion

In the previous sections we have considered many formulations for the same variation of RSA. A first characteristic to take into account is the model size,

i.e., the number of variables and constraints employed by each formulation. Table 1 shows the number of variables and constraints of the formulations presented in Section 2. We show these values associated to the families of models, since all the models within the same family have the same order of magnitude for variables and constraints.

Family	Variables	Constraints
Demand-Range (DR)	$\Theta(\|D\|^2)$	$\Theta(\|D\|^2\|E\|)$
Demand-Slot (DS)	$\Theta(\|D\|S)$	$\Theta(\|D\|^2\|E\|S)$
Demand-Slot-Link (DSL)	$\Theta(\|D\|\|E\|S)$	$\Theta(\|D\|\|E\|S)$
Demand-Range-Link (DRL)	$\Theta(\|D\|\|E\|S)$	$\Theta(\|D\|\|E\|S)$

Table 1: Number of variables and constraints of the formulations presented in Section 2

The number of constraints associated to the family DS makes quite impractical to generate ILP models with large instances. As an example, with $S = 100$, $|D| = 100$ and $|E| = 10$, we are trying to handle a set of approximately 10 millions of constraints, while with the families DR, DSL, DRL, we are talking about sets of approximately 100.000 constraints. In section 3 we solve instances of different sizes with the models of the family DS in order to show a detailed behavior of all the formulations. The sizes of the sets of variables are not so different between the different families as with the sets of constraints. With the same configuration as above, DR and DS formulations have approximately 10.000 variables, while DSL and DRL have approximately 100.000 variables. As we will see in the following sections, these values associated to the families influence in the model generation, in some cases with critical consequences.

It is not possible in general to perform a theoretical analysis concerning the practical performance of an ILP model, and this motivates the empirical analysis in Section 3. Although the model size is not a good predictor of the performance of an ILP solver, if a model has a very large number of variables or constraints then it may be difficult or impossible to generate for medium- or large-sized instances.

Nevertheless, it is usual to consider the strength of the linear relaxation compared to the convex hull of feasible solutions. In this sense, constraints (14) are the well-known *flow conservation constraints* and, isolated, generate an integer convex hull, a very desirable property in this kind of models. This fact motivated us to look for formulations featuring this kind of constraints. Although not generating integer convex hulls, some constraints are known to provide stronger relaxations than alternative versions as, e.g., constraints (24) versus constraints (23) in models DR-SC and DR-BSA, respectively, or constraints (30) versus constraints (28) in models DS-ACC and DS-BF, respectively. In these cases, the models with stronger relaxations are more appealing from a theoretical perspective.

3 Computational experiments with ILP formulations

In this section we report the results of our experimental comparison of the models presented in Section 2. On the one hand, Section 3.1 systematically explores the performance of the models for different parameters generating the number of slots and the number of demands. Each ILP model was tested with eight network topologies taken from the optical fiber literature, using different combinations of demands and slots sets. On the other hand, Section 3.2 reports computational results on realistic instances.

The models were implemented in OPL and solved with IBM ILOG CPLEX 12.6, on a 2.10 GHz Intel core i5 computer.

Network topologies

Each network topology is given by a graph $G_T = (V_T, E_T)$ where the set of vertices V_T and edges E_T are the set of nodes and links of the network, respectively. The information of each graph $G_T = (V_T, E_T)$, with T in the set of fiber optical network topologies, can be seen in Table 2. The last column of this table denotes the sparsity coefficient of the graph G_T and is given by $Sparsity_T = |E_T|/EdgesComplete_T$, where $EdgesComplete_T = (|V_T| * (|V_T| - 1))/2$ (i.e., the amount of edges of the complete graph).

Topology (T)	Nodes (V_T)	Edges (E_T)	$Sparsity_T$
British Telecom	20	32	0.16
COST239	11	26	0.47
Deutsche Telekom-14	14	23	0.25
Deutsche Telekom-21	21	32	0.15
EON	19	35	0.20
NSF	14	21	0.23
Spanish Telefonica	21	34	0.16
UKNET	21	39	0.18

Table 2: Number of nodes and edges and sparsity coefficient of each network topology.

3.1 Exploration of the parameter space

In order to compare the performance of our ILP formulations, we evaluate them using a big set of generated instances. For each of the network topologies above presented we use the values $\{10,\ 20,\ 40\}$ and $\{2 * |D|,\ 2 * |D| + 10,\ 2 * |D| + 20,\ 2 * |D| + 30\}$ for $|D|$ and S, respectively. Each $d \in D$ has an associated volume, generated with uniform distribution from $\{1, \ldots, S\}$, and an associated source and destination, generated with uniform distribution

in V. Then, an instance is defined by a specific network topology, a set of demands to satisfy, and a number of available slots which depends on the number of demands.

We show the results of that vast set of experiments in a summarized way in Table 3. The first column corresponds to the ILP formulation. Due to the big number of tests we have done in this subsection -approximately 1200 runs-, *Time limit* is set to 600 seconds so if the optimal solution is not reached within that time, the execution is stopped. If the size of the generated model is such that CPLEX cannot handle it, an *out of memory* exception is reported and we jump to the next experiment. Also, since the volume, source, and destination of demands are randomly generated, the created instance may be *infeasible*. The second, third, and fourth columns of Table 3 show the percentage of instances solved to optimality, of instances not solved in the proposed time, and of instances where an *out of memory* exception was reached, respectively, for each formulation. *Infeasible* instances are not taken into account when these percentages are calculated. The last column corresponds to the bigger size of $|D|$ solved to optimality.

Formulation	% Solved	% Time Limit	% Out of memory	Solved Size Limit
DR-BF	86	14	0	40
DR-OSB	85	15	0	40
DR-AOV	90	10	0	40
DR-BSA	37	4	59	20
DR-SC	22	0	78	10
DS-BF	34	18	48	10
DS-ACC	38	3	59	20
DS-SCC	46	2	52	20
DSL-BF	46	52	2	20
DSL-ASCC	17	17	64	10
DSL-ASB	21	32	47	10
DRL-BF	35	9	56	10

Table 3: Summarized results for the presented ILP formulations.

In order to show a more exhaustive analysis, we choose the formulations with the best performance in their respective families and present a detailed behavior of them. We do not take a representative of the family DRL because of their similarities in terms of the characteristics to be evaluated with the DSL family. The formulations DR-AOV, DS-SCC and DSL-BF were chosen and Figures 4, 5 and 6 compare their average of variables, constraints and solving times, respectively. The results are shown according to the sizes of the instances (combination of $|D|$ and S), and the average is performed over all the instances, i.e., the sum is calculated over all the instances of each size divided by the total number of instances of each size. The model sizes, represented by their numbers of variables and constraints, are shown in a logarithmic scale and as we can see they are exponential in the instance size.

In these figures, the bar associated with the formulation DS-SCC no longer appears starting at instance $(40, 90)$, since the formulations could not be generated due to their size. The same happens with the formulation DSL-BF starting at instance $(80, 160)$.

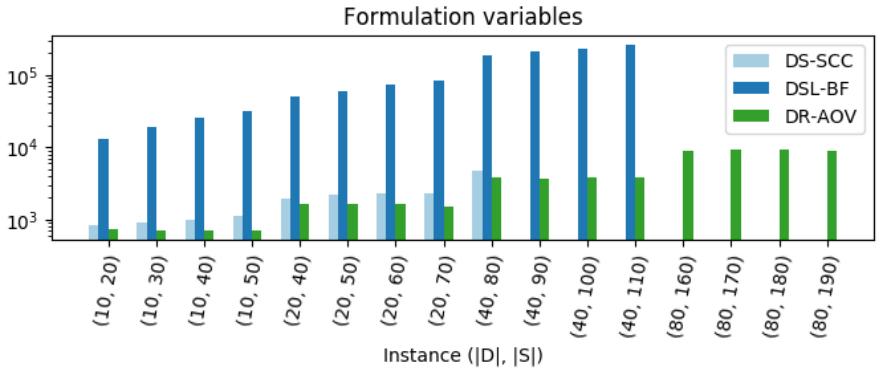

Fig. 4: Average number of variables for Models *DS-SCC*, *DSL-BF* and *DR-AOV*.

Figure 4 shows that Model DSL-BF has a large number of variables. This is due to the way in which the assignment is represented in the family of models this formulation belongs to. As in the rest of the models of the DSL family, a set of variables is used for the assignment of demands to links and slots, which has a size equal to $|D||E|S$. The other model families represent on the one hand the assignment of demands to links and on the other the assignment of demands to slots, so the number of variables is considerably less. In the case of Model DS-SCC, we find a theoretical number of variables equal to $|D|(|E| + S)$ which is smaller than the DR-AOV's amount which is equal to $|D||E| + 2|D| + |D||D|$.

Figure 5 shows the remarkable big number of constraints Model DS-SCC contains. This is due to the family of constraints added in order to ensure the non-overlapping property, which asks that each pair of demands is not assigned the same slot at any link ($|D||D||E|S$ constraints). This large number of constraints, which is one order of magnitude bigger than the amount of variables of the DSL-BF, causes from a certain number of demands and slots (size of set $|D|$ and value of S), the associated formulations cannot even be generated by CPLEX, raising *out of memory* exceptions.

In both aspects, Model DR-AOV was the best model in our first evaluation stage. In terms of the number of variables, a difference that can reach up to two orders of magnitude comparing with Model DSL-BF can be seen and a small difference comparing with Model DS-SCC. It should be noted that as we are using integer variables for the assignment of demands with slots, the number of variables does not increase as S grows. In terms of the number of

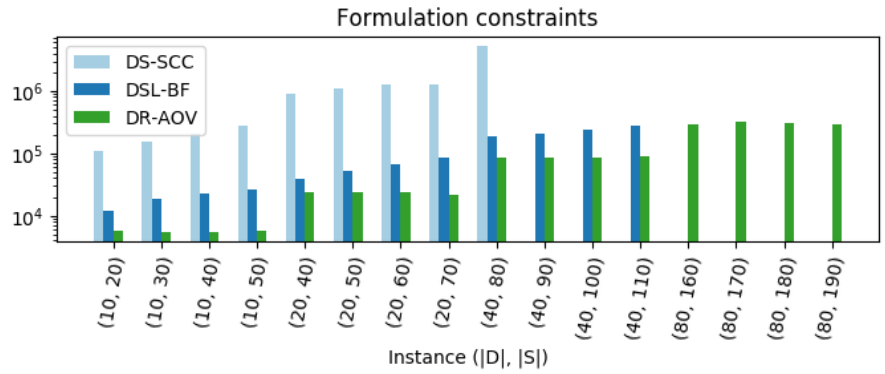

Fig. 5: Average number of constraints for Models DS-SCC, DSL-BF and DR-AOV.

constraints, it is possible to see a difference that can reach up to two orders of magnitude comparing with the Model DS-SCC and of one order of magnitude comparing with Model DSL-BF. As with the number of variables, the number of constraints is not altered by modifying S, this is because no constraint depends on this value.

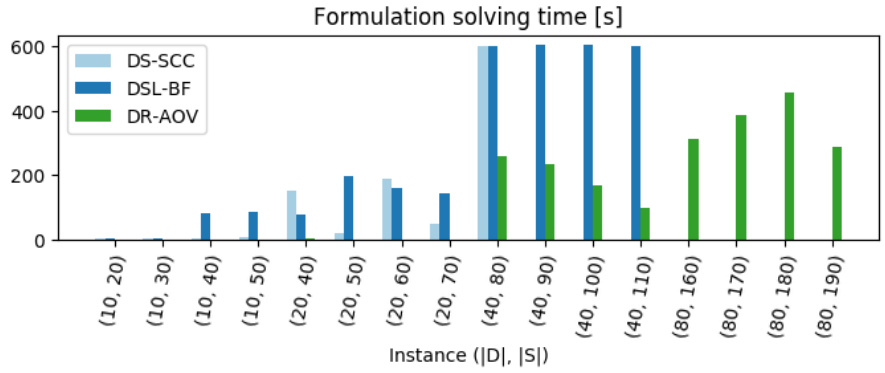

Fig. 6: Average solving time for Models DS-SCC, DSL-BF and DR-AOV.

Finally, as we can see in Figure 6, Model DR-AOV is the only model that could solve all the instances and the one with the best execution time. Model DS-SCC was the model with the least number of instances solved to optimality. This is because, on the one hand, as it has already been mentioned, formulations cannot even be generated for many instances due to their size. On the other hand, Model DSL-BF has a smaller number of instances not

solved to optimality but this is because of *time limits* exceptions, which can be substantially reduced by setting a higher maximum resolution time.

The e-companion to this manuscript (Online Resource 1) contains detailed results of all the experiments performed in this section.

3.2 Performance comparison in real scenarios

Along this section we will show the results of computational experiments we have done with all the formulations of Section 2 in real scenarios. We use British Telecom and NSF as topologies network for the instances, trying to cover different sparsity coefficients. We consider slots of 12.5GHz and two optical spectrum widths: 400GHz and 4000GHz, which could be generated using a laser diode and LED as optical source, respectively. For this second stage of computational experiments, S could be 32 or 320. For each element in the demand set, the requested bandwidth is 10, 40, or 100Gbps. That is, for each $d \in D$, the associated volume v_d was generated with uniform distribution from $\{1, 2, 4\}$, and the associated source and destination was generated with uniform distribution in V. The length of D depends on the topology and S: for the network NSF and S equal to 32, we use 30, 50 and 80 as values for $|D|$ while with S equal to 320, we use the values 100, 150 and 180; for the $British$ network the values used for $|D|$ will be 30, 50 and 70 when S is 32, and 30, 50 and 70 when S is 320. We use different values for $|D|$ depending on the topology and S in order to keep the difficulty and size under control. On one hand, instances generated with $British$, $S = 32$ and $|D| = 80$ are not possible to solve for any model, even using 1 hour as time limit. On the other hand, NSF has 14 nodes so we could not generate more than 182 demands without repeating demands.

Tables 4 and 5 show the performance of the formulations in experiments that use NSF and British as network topology, respectively. For this second stage of evaluation we set *Time limit* to 1800 seconds, so if a resolution exceeds this bound the label *Time* is used. Also, if it is not possible for CPLEX to generate the formulation due to the instance size, we use the character "-". Again, the formulations BF, OSB, and AOV of the family DR beside DSL-BF present the best performance, this time taking into account the solving times and the size of the instances that can be handled without getting a memory exception. For all the formulations of the family DS and most of DSL, CPLEX cannot generate the models to be solved. As we see in Table 1, this is because of the big set of constraints this formulations constain.

Comparing with the formulations taken from the literature and under this set of experiments, most of our formulations present better solution times and could deal with bigger instances. For NLS, with NSF as topology and S equal to 32, the results of the experiments are almost the same as the ones from our formulations. However, with S equal to 320 it is not possible for CPLEX to generate the ILP models for all the sizes of D, since the non-overlapping constraints have a large number of elements in this case. As an example and

as remarked before, with $S = 320$, $|D| = 100$ and $|E| = 30$, we are trying to handle a set of approximately 100 millions of constraints. For NL-CA, only for two instances could we get an optimal solution, in all the other cases a time out was reached.

Formulation	$S = 32$			$S = 320$		
	$\|D\| = 30$	$\|D\| = 50$	$\|D\| = 80$	$\|D\| = 100$	$\|D\| = 150$	$\|D\| = 180$
DR-BF	1	5	18	21	100	109
DR-OSB	2	6	18	30	76	296
DR-AOV	1	3	6	13	42	365
DR-BSA	33	129	594	-	-	-
DR-SC	51	179	-	-	-	-
DS-BF	17	50	-	-	-	-
DS-ACC	20	65	-	-	-	-
DS-SCC	14	49	-	-	-	-
DSL-BF	3	4	22	243	Time	Time
DSL-ASCC	18	29	161	-	-	-
DSL-ASB	11	21	Time	-	-	-
DRL-BF	13	26	104	Time	Time	Time
NLS	12	46	162	-	-	-
NL-CA	24	64	Time	Time	Time	Time

Table 4: Computational times of ILP formulations in realistic experiments with NSF as network topology, in seconds. (Nodes: 14, Edges: 21)

Formulation	$S = 32$			$S = 320$		
	$\|D\| = 30$	$\|D\| = 50$	$\|D\| = 70$	$\|D\| = 100$	$\|D\| = 150$	$\|D\| = 200$
DR-BF	2	11	261	51	285	923
DR-OSB	2	9	19	81	349	Time
DR-AOV	2	5	9	23	60	122
DR-BSA	64	1200	Time	-	-	-
DR-SC	201	Time	Time	-	-	-
DS-BF	51	335	Time	-	-	-
DS-ACC	37	1148	Time	-	-	-
DS-SCC	65	362	Time	-	-	-
DSL-BF	10	27	151	Time	Time	Time
DSL-ASCC	48	72	Time	-	-	-
DSL-ASB	59	240	Time	-	-	-
DRL-BF	27	50	656	Time	Time	-
NLS	26	285	Time	-	-	-
NL-CA	Time	Time	Time	Time	Time	Time

Table 5: Computational times in seconds of ILP formulations in realistic experiments with British Telecom as network topology. (Nodes: 20, Edges: 32)

4 Conclusion

In this work we have presented several integer programming models that solve the same version of RSA. We have tried with different families of variables and constraints, obtaining mixed results.

It is important to note that for our formulations we did not take into account the common two-slot guard between channels. It is easy to solve real instances by just adding a slot to each side of each demand and doing the same to the channels (i.e., if we have S available slots per channel, we allow S+2 slots). Also we did not take into account the problems related to distance (trade-off modulation format adaptation), assuming that all of them are going to be taken into account when setting the topology and the amount of available slots per channel. Without loss of generality we used the same amount of slots for every channel. Fixing that amount with the maximum value, given the case in which there were channels with less capacity, just adding restrictions that enforce to be set the last not existent slots we get a feasible solution.

As the computational tests were made using OPL and the solver as a black box, the experiments of Section 3 give for each model an approximate idea of the instance size that it is able to solve without more sophisticated integer linear programming techniques. Our computational experiments, both real and random, suggest that Models DR-BF, DR-OSB and DR-AOV outperform the other formulations, solving almost 90% of the proposed instances without memory issues and getting optimal values over the biggest instances even one order of magnitude faster than the others. This may be due to the way in which the slot and link assignments are represented in these formulations, namely the assignment of slots to demands is implicitly represented. In these models, an integer variable indicates the first slot assigned to a demand and, being its volume a fixed value, we know which other slots will be assigned to the demand as well. Another factor that can contribute to the good performance of these formulations is the "ordering" that is established between the demands in order to avoid slot overlappings.

As an ongoing work, we are currently exploring the combinatorial and geometrical structures of the polytopes associated with the most promising formulations, in order to gain knowledge that might be useful in cutting-plane environments.

References

1. S. Altinakar, G. Caporossi, and A. Hertz, *A comparison of integer and constraint programming models for the deficiency problem.* Computers & Operations Research 68 (2016) 89-96.
2. K. Christodoulopoulos, I. Tomkos, and E. A. Varvarigos, *Elastic bandwidth allocation in flexible OFDM-based optical networks.* IEEE J. Lightw. Technol., vol. 29, no. 9, pp. 1354–1366, 2011.
3. M. Jinno *Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network* IEEE Communications Magazine, vol. 48, no. 8, pp. 138-145, 2010.

4. M. Klinkowski and K. Walkowiak, *Routing and spectrum assignment in spectrum sliced elastic optical path network.* IEEE Commun. Lett., vol. 15, no. 8, pp. 884–886, 2011.
5. C. Moreno-Camacho, J. Montoy a-Torres, and M. Vélez-Gallego, *A comparison of mixed-integer linear programming models for workforce scheduling with position-dependent processing times.* Engineering Optimization 50-6 (2018) 917-932.
6. O. Koné, C. Artigues, P. Lopez, and M. Mongeau, *Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources.* Flexible Services and Manufacturing Journal 25/1-2 (2013) 24-47.
7. L. Velasco, M. Klinkowski, M. Ruiz, and J. Comellas, *Modeling the routing and spectrum allocation problem for flexgrid optical networks.* Photonic Netw. Commun., vol. 24, pp. 177–186, 2012.
8. L. Velasco, M. Klinkowski, M. Zotkiewicz, and M. Pioro, *Optimization models for flexgrid elastic optical networks*, proceedings of the 15th International Conference on Transparent Optical Networks (ICTON 2013), Cartagena, Spain, 23 - 27 June 2013.
9. Y. Wang, X. Cao, Q. Hu, and Y. Pan, *Towards elastic and fine-granular bandwidth allocation in spectrum-sliced optical networks.* IEEE J. Opt. Commun. Netw., vol. 4, no. 11, pp. 906–917, 2012.

Appendices

A Consecutive ones proposition

Fix a demand and a link used by this demand. Let $S \in \mathbb{N}$ be the amount of available slots. If $s \in \{1, \ldots, S\}$ is a slot then call X_s the binary variable equal to one if and only if the slot s is used by the demand. Fix the demand volume h such that $h \leq S$ and consider the following constraints:

$$\sum_{\substack{s \in \{1,\ldots,S\}: \\ s \equiv j \ (mod\ h)}} X_s \ = \ 1 \qquad \forall j \in \{1, \ldots, h\} \qquad (52)$$

$$\sum_{\substack{s \in \{1,\ldots,i\}: \\ s \equiv i \ (mod\ h)}} X_s \ \geq \ \sum_{\substack{s \in \{1,\ldots,i-1\}: \\ s+1 \equiv i \ (mod\ h)}} X_s \qquad \forall i \in \{1, \ldots, S+1-h\} \qquad (53)$$

Theorem 1 *Constraints (52) and (53) ensure that there are exactly h consecutive variables in $\{X_1, \ldots, X_S\}$ taking value 1.*

Proof Using constraints (53) and a property of constraints (52) we shall prove that there are at least h consecutive variables taking value 1, and using (52) we shall prove that those h variables are the only ones taking this value.

For $i = 1, \ldots, s+1-h$, let C_i be the constraint (53) associated to the index i, and let LHS_i and RHS_i be the left-hand and the right-hand side of C_i respectively, i.e.,

$$C_i \ : \ LHS_i \ \geq \ RHS_i.$$

Since $LHS_i = RHS_{i+1}$ for $i = 1, \ldots, s-h$, then C_i takes the form

$$LHS_{i+1} \geq LHS_i. \tag{54}$$

Let L_i be the set of indices of LHS_i, such that X_s appears in LHS_i if and only if $s \in L_i$. We have $i \in L_i$ but $i \notin L_{i+1}, \ldots, i \notin L_{i+h-1}$, implying that i is the largest element in L_i, i.e., $i = \max(L_i)$. For $q = 0, \ldots, h-1$ and $i = 1, \ldots, S-q$, we conclude that the only index larger than i in L_{i+q} is $i+q$, i.e.,

$$\{i' : i' \geq i,\ i' \in L_{i+q}\} = \{i+q\} \tag{55}$$

Now we use these results to prove that any feasible solution that satisfies constraints (52) and (53) has at least h consecutive variables set to 1. For $i = 1, \ldots, S$, we have

$$\{s \in \{1, \ldots, i\} :\ s \equiv i\ (mod\ h)\} \subseteq \{s \in \{1, \ldots, S\} :\ s \equiv i\ (mod\ h)\}.$$

Thus, constraints (52) imply:

$$LHS_i \leq 1 \tag{56}$$

for $i = 1, \ldots, S+1-h$.

Let $X^* \in \{0,1\}^S$ be a feasible solution, and let p be the lowest index such that $X^*_p = 1$. Let i^* be the lowest index such that $X^*_p \in LHS_{i^*}$. Because of (56) and since $x_p = 1$ then $LHS_{i^*} = 1$. Likewise, because of (54) and (56), we have $LHS_k = 1$ for every $k \geq i^*$. In particular,

$$LHS_k = 1 \text{ for } k = i^*, \ldots, i^* + h.$$

By the definition of p, we have $X_{p'} = 0$ for $p' < p$. This implies that in LHS_k such that $i^* < k < i^* + h$ only variables with indices greater than p can take value 1; but because of (55) those are $\{X_{p+1}, \ldots, X_{p+h-1}\}$.

Therefore, we have at least h consecutive variables taking value 1. Consider now the union of the sets $\{s \in \{1, \ldots, S\} :\ s \equiv j\ (mod\ h)\}$ for $j \in \{1, \ldots, h\}$. Since all their elements are disjoint, then

$$\bigcup_{j=1}^{h} \{s \in \{1, \ldots, S\} :\ s \equiv j\ (mod\ h)\} =$$

$$\{s \in \{1, \ldots, S\} :\ s \equiv j\ (mod\ h),\ j \in \{1, \ldots, h\}\} = \{1, \ldots, S\}.$$

This implies that there are at most h variables taking value 1, i.e.,

$$\sum_{j \in \{1,\ldots,h\}} \sum_{\substack{s \in \{1,\ldots,S\}: \\ s \equiv j\ (mod\ h)}} X_s \leq h$$

Since there are at least and at most h consecutive variables taking value 1, there are exactly h variables taking value 1, hence concluding the proof.