A Branch-and-Price Algorithm for the Design of Survivable IP-over-Optical Networks

Virginie Gabrel, A. Ridha Mahjoub, Raouia Taktak

LAMSADE, Université Paris Dauphine,
Place du Maréchal de Lattre de Tassigny 75775 Paris Cedex 16, France

Eduardo Uchoa

Departamento de Engenharia de Produção, Universidade Federal Fluminense.
Rua Passo do Pátria, 156 Bloco E Sala 440, Niterói 24210-240, RJ, Brasil

Abstract

In this paper, we consider a multilayer survivable network design problem which arises in telecommunications. Assume given an IP-over-optical network and a set of demands in the IP layer such that for each demand we know two node-disjoint paths routing it in this layer. The problem consists in finding, for every demand, two node-disjoint paths routing it in the optical layer. These paths must go through the optical switches corresponding to the routers visited in the IP-layer in the same order as the routers in the IP layer. We show that the problem is NP-hard and propose integer programming formulations. Moreover, we devise a Branch-and-Price algorithm. Extensive computational results are presented, showing the efficiency of our algorithm.

Keywords: IP-over-optical networks, survivability, integer programming, Branch-and-Price algorithm, pricing problem, column generation.

1. Introduction

In the two past decades, telecommunication networks have seen a big development with the advances in optical technologies and the explosive growth of the

Email addresses: gabrel@lamsade.dauphine.fr (Virginie Gabrel), mahjoub@lamsade.dauphine.fr (A. Ridha Mahjoub), taktak@lamsade.dauphine.fr (Raouia Taktak), uchoa@producao.uff.br (Eduardo Uchoa)

Internet. Thanks to the optical systems, the data traffic has increased dramatically and network capacities have been almost unlimited allowing, thus, the transfer of huge quantities of many varieties of information (voice, video,...). Hence, in the event of a catastrophic failure, a big amount of traffic may be lost. Consequently, telecommunication operators have constantly cared about the survivability of their networks. In this context, a network is said to have a *survivable topology* if this latter permits the service to be restored and the network to remain functional in the event of a failure.

Moreover, due to the last evolution of telecommunication networks' architecture, traffic data have been analysed, described and managed in a multilayer structure. Indeed, it is quite natural to assume that the more elaborated functionalities of a network rely on a set of simple ones provided by some lower layer. This is in particular the case of modern telecommunication networks where different technologies (SDH/SONET, WDM, Gigabyte Ethernet, ATM, IP,...) are combined in various ways resulting on successive technological layers. From a practical point of view, this means that in order to carry its traffic on some layer, the network may need the use of a lower-level's technology. Then, several layers can be piled up in order to have an operational network offering a variety of services. The advantage of this is that each technology can be used for its most favorable features. The drawback, however, is that each technology, and hence each layer, manages its own routing control scheme independently from the others, and addresses its own survivability issues. As a consequence, in a network design problem, reliability had been considered layer by layer without tackling the redundancy and the non-optimality yielded by the multilayer structure. The introduction of new protocols in telecommunications (like GMPLS [1]) has given a new trend for multilayer data networks. This new system provides a common signaling and routing framework between the different layers, and it does not restrict the way these layers work together. This evolution is yielding new survivability issues in multilayer networks.

In this paper we consider a multilayer survivable network design problem that has application in IP-over-optical networks. Assume given an IP-over-

optical network and a set of demands in the IP layer such that for each demand we know two node-disjoint paths routing it in the IP layer. The problem consists in finding, for every demand, two node-disjoint paths routing it in the optical layer. These paths must go through the optical switches corresponding to the routers visited in the IP-layer, and respect the order of these routers. We give integer programming formulations for this problem. Moreover, we present a Branch-and-Price algorithm to solve it, along with substantial computational results.

The problem of designing survivable multilayer networks has been widely studied in the literature. Dahl et al. [2] were the first to be interested in this problem. Many variants have later been studied and several methods of resolution have been devised. In particular, exact methods have been proved to be efficient for solving to optimality large-scale problems in the multilayer context. Borne et al. [3] study the problem of securing the IP layer given failure scenarios in the optical layer in IP-over-optical networks. They give a 0-1 integer programming formulation for the problem and study the associated polytope. They also identify new classes of valid inequalities and describe conditions for these inequalities to be facet-defining. Then, they discuss separation techniques and describe a Branch-and-Cut algorithm that is tested on random and realistic instances. In [4], Belotti et al. consider the problem of two-layer networks design taking into account the impact of statistical multiplexing in the MPLS layer. They propose a path-based Mixed Integer Program for the problem. A column generation approach coordinated with a Lagrangian relaxation has been introduced to solve the model. Later, Gouveia et al. [5] consider the hop-constrained node survivable network design problem in the context of MPLS-over-WDM networks. The problem is considered within two different survivability mechanisms: path diversity and path protection. An integer linear programming model is proposed for both mechanisms and a cost analysis of the design solutions is handled on the NSFNet and EON real world networks. Further optimization aspects are considered by Orlowski et al. [6]. The authors study the problem of planning multilayer SDH/WDM networks. The goal is to

find a minimum cost installation of links and nodes hardware for both layers, subject to many practical engineering constraints as well as constraints related to survivability against failures. A mixed integer programming formulation is proposed and solved using an efficient Branch-and-Cut algorithm.

Later, Orlowski and Pióro [7] focus more on survivability issues. They survey different mechanisms of survivability in telecommunication networks. Several path-based survivability mechanisms are considered and the complexity of the corresponding pricing problems is studied. In [8], Raghavan et al study the problem of designing logical and physical layers in multilayer networks with non-bifurcated traffic flows. Unlike classical works dealing with the two layers independently or in a sequential way, the authors proposed to simultaneously design the two layers of the graph. A Branch-and-Price algorithm that solves simultaneously the logical topology design and the traffic routing in the optical layer is considered.

This paper is organized as follows. In the following section, we discuss the IP-over-optical networks and examine the interconnection models proposed for these networks. In Section 3, we present the multilayer survivable network design problem considered in this paper, called the Multilayer Survivable Optical Network Design problem and give some notations. In Section 4, we study the complexity and show that the problem is NP-hard even when only one demand is considered. In Section 5, we give two integer programming formulations. The first one is given in terms of routing variables and the second is a path based formulation. In Section 6, we present a Branch-and-Price algorithm devised to solve the path formulation. We discuss the corresponding pricing problem and prove that this reduces to a shortest path problem and can then be solved in polynomial time. We also devise a branching scheme that preserves the tractability of the pricing problem, and describe a primal heuristic. A computational study is presented and discussed in Section 7. The paper is concluded by some remarks given in Section 8.

2. Multilayer infrastructure for telecommunication networks

Telecommunication networks are now moving toward a model of high-speed routers interconnected by intelligent optical core networks. Moreover, there is a general consensus that the control plan of the optical networks should utilize IP-based protocols for dynamic provisioning and restoration of lightpaths [9, 10]. The optical network consists of optical switches (also called Optical Cross-Connects (OXC)) interconnected by optical links. The IP and optical networks communicate through logical control interfaces called User-Network-Interfaces (UNI). The optical network essentially provides point-to-point connectivity between routers in the form of fixed bandwidth lightpaths. These lightpaths define the topology of the IP network.

Each router in the IP network is connected to at least one of the optical switches. Moreover, to each link in the IP network between two routers corresponds a routing path in the optical layer between two switches corresponding to these routers. Figure 1 shows an IP-over-optical network. The IP network contains four routers denoted by R_1, R_2, R_3 and R_4. The optical network holds seven optical switches denoted S_1 to S_7. Optical switches S_1, S_2, S_3, S_4 correspond to the routers R_1, R_2, R_3, R_4, respectively. Each optical switch communicates with one router through the interfaces UNI.

In order to formalize the IP-over-optical network, the Internet Engineering Task Force (IETF) proposed three interconnection models [11] : the overlay, peer and augmented models.

The overlay model is the model currently used by the operators. Under this model, the IP network routing, topology distribution and signaling protocols are independent from the corresponding protocols in the optical network. The client network requests a connection between two routers. The optical network offers end-to-end wavelength services to client network via the UNI to respond to this request. In this model, the client network has otherwise no control over the exact routing and priority received within the intelligent optical network which has full control over its network resources. The advantage of the overlay

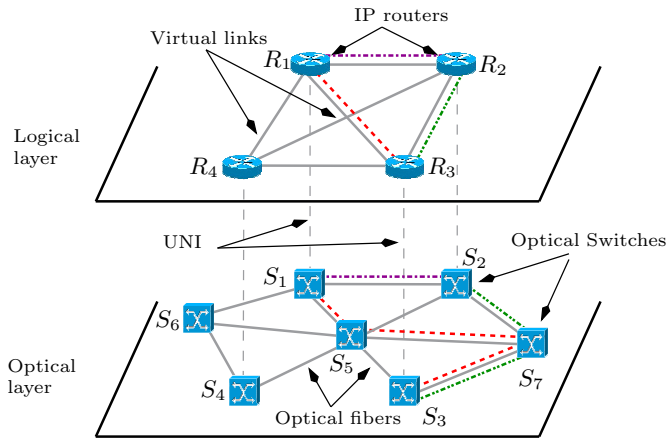

Figure 1: Example of multilayer IP-over-Optical network

model is that it is the most practical for near-term deployment. Its drawback is that it requires the creation and management of IP routing adjacencies over the optical network.

Under the peer model, the IP and optical networks are treated together as a single integrated network managed and traffic-engineered in a unified manner. In this regard, the optical switches are treated just like any other IP router as far as the control plane is concerned. IP routers and optical switches use the same addressing scheme. The optical network elements become IP addressable entities. The optical network topology is fully visible to routers. A single routing protocol instance runs over both the IP and optical domains. The advantage of the peer model is that it allows seamless interconnection of IP and optical networks. The architecture is scalable, functionality is not duplicated and conflicts between several control planes do not arise. Its drawback is that it requires routing information specific to optical networks to be known to IP routers. There are thus excessive information flows between the two networks. This type of tight integration may not be practical in the near term. Despite its drawbacks, the peer model can be expected to be the architecture to be adopted in the long term if indeed IP dominates the scene.

The augmented model is between the overlay and the peer models. There are separate routing instances of the same routing protocol in the IP and optical domains, but with limited routing exchange between the two domains. Some reachability information is exchanged between the two networks but the topology of the optical network is opaque to the client network. This option may be a good compromise as it is relatively easy to deploy in the near term compared to the peer model, and at the same time it is less rigid and more efficient than the overlay model.

The introduction of the peer model (and the GMPLS control plane) gives rise to new survivability issues for the IP-over-optical networks. Consider the IP-over-optical network given in Figure 1. (R_1, R_2), (R_2, R_3) and (R_1, R_3) represent virtual links in the logical layer. These links are ensured by lightpaths in the optical layer. Figure 1 shows that (R_1, R_2) is physically routed by (S_1, S_2), (R_2, R_3) by (S_2, S_7, S_3) and (R_1, R_3) by (S_1, S_5, S_7, S_3). Note that the network is not survivable. In fact, if the node S_7 in the optical layer breaks down, the physical paths from S_1 to S_3 and from S_2 to S_3 are interrupted. Consequently, the virtual links (R_1, R_3) and (R_2, R_3) in the logical layer will break down as well.

In consequence, survivability strategies have to be considered. If the transport network is fixed, one has to determine the suitable client topology for the network to be survivable. If however, the client network is fixed as well as its routing, then we have to determine an optimal survivable routing in the transport network. And finally if none of both layers is fixed, then a routing has to be determined for each of the layers in order to get a whole survivable network. In this paper we shall be interested in the second case. That is, we will consider the problem of, given a secure topology for the logical layer, finding a corresponding reliable topology in the optical layer.

3. The Multilayer Survivable Optical Network Design Problem

3.1. The problem

Consider an IP-over-optical network. Consider a set of demands between pairs of routers of the IP-layer such that for each demand we know two node-disjoint paths routing it in the virtual layer. Suppose that with each link of the physical layer is associated a cost associated to its realization. The Multilayer Survivable Optical Network Design (MSOND) problem consists in designing a minimum-cost optical network such that each demand is routed through two node-disjoint physical paths. Moreover, these paths must go in order through the optical switches corresponding to the routers visited in the virtual paths.

In order to illustrate this, consider the graphs of Figure 2 which correspond to the multilayer network of Figure 1. Graph G' represents the logical layer and its node set is $V' = \{v_1, v_2, v_3, v_4\}$. Graph G represents the optical layer and its node set is $V = \{w_i, i = 1, ..., 7\}$. To each node $v_i \in V', i = 1, ..., 4$ corresponds a node $w_i \in V, i = 1, ..., 4$. Nodes w_5, w_6 and w_7 of graph G do not have correspondent nodes in graph G'.

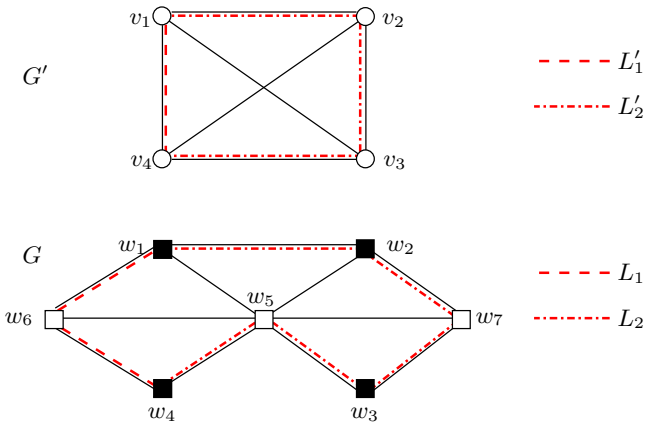

Figure 2: An example of a feasible solution for the MSOND problem

In G', a demand between nodes v_1 and v_4 is routed by the two node-disjoint paths $L'_1 = (v_1, v_4)$ and $L'_2 = (v_1, v_2, v_3, v_4)$. In G, a feasible solution for

the problem is given. The first path is routed by $L_1 = (w_1, w_6, w_4)$, and the second path by $L_2 = (w_1, w_2, w_7, w_3, w_5, w_4)$. Note that paths L_1 and L_2 are node-disjoint. Also note that L_1 and L_2 are respecting the order of the nodes visited in L'_1 and L'_2. In fact, L_2 is visiting the terminal w_2 corresponding to v_2 before passing through the terminal w_3 corresponding to v_3 reaching at the end the destination w_4 corresponding to v_4. The nodes w_1, w_2, w_3 and w_4, represented by filled squares, corresponding to v_1, v_2, v_3 and v_4, the nodes visited in L'_1 and L'_2 in G', are called *terminals*. The nodes in $V \setminus \{w_1, w_2, w_3, w_4\}$ are called *Steiner nodes* or *steiners*.

Note that, since the graph is undirected, looking for the two paths L_1 and L_2 for the demand of Figure 1 is equivalent to searching an elementary cycle going through the terminals w_1, w_2, w_3 and w_4 in that specific order. This shows that the MSOND problem has a relationship with the well-known Traveling Salesman Problem (TSP). In particular, the MSOND problem for a single demand can be seen as a Steiner TSP with an order constraint on its terminals. This later problem can be presented as follows. Given a graph $G = (V, E)$ with edge costs and a set of terminal nodes $W \subseteq V$, the *Steiner TSP* consists in finding a minimum-cost tour going through the terminals W. A particular interest has been paid to the Steiner TSP. Exact methods and polyhedral studies are proposed in [12], [13], [14]. The complexity aspect of the problem is also studied and several approximation results are given in [15].

3.2. Definitions and notations

We consider undirected and finite graphs. An undirected graph will be denoted by $G = (V, E)$ where V is the *node set* and E is the *edge set* of G. Given $W \subseteq V$, we denote by $\delta_G(W)$ the set of edges of E having exactly one node in W. The edge set $\delta_G(W)$ is called a *cut*. If $W \subset V$, $\overline{W}$ denotes $V \setminus W$.

The MSOND problem can be presented in terms of graphs as follows. We associate with the logical layer an undirected graph $G' = (V', E')$ where nodes correspond to the routers and edges to possible links between these routers. We associate with the optical layer an undirected graph $G = (V, E)$ where nodes

correspond to the optical switches and edges to the possible physical links between these switches. With every router $v_i \in V'$ is associated an optical switch $w_i \in V$. We assume that between nodes of G' there exist traffic demands. Denote by K the set of these demands. For $k \in K$, denote by (O'_k, D'_k) the pair of routers origin-destination of demand k and by O_k and D_k the corresponding optical switches in the optical layer. For $k \in K$, let $L'_{k,1} = (O'_k = v_1^{k,1}, ..., v_j^{k,1}, ..., v_{l'_{k,1}}^{k,1} = D'_k)$ and $L'_{k,2} = (O'_k = v_1^{k,2}, ..., v_j^{k,2}, ..., v_{l'_{k,2}}^{k,2} = D'_k)$ be the two paths routing demand k in graph G', where $l'_{k,1}$ and $l'_{k,2}$ represent the lengths of $L'_{k,1}$ and $L'_{k,2}$, respectively. Let $T_k = (w_1^k = w_1^{k,1} = O_k, \ldots, w_{l'_{k,1}}^k = w_{l'_{k,1}}^{k,1} = D_k, w_{l'_{k,1}+1}^k = w_{l'_{k,2}-1}^{k,2}, \ldots, w_{l'_{k,1}+l'_{k,2}-1}^k = w_2^{k,2})$ be be the ordered set of terminals of G corresponding to the nodes of $L_{k,1}$ and $L_{k,2}$. The nodes in T_k are the terminals of demand k, the set of Steiner nodes of k is defined as $S_k = V \backslash T_k$. In the case of a single demand, we simply denote by T the set of terminals of the demand and by S the set of its Steiner nodes.

The cycle that must route the terminals in T_k, for some $k \in K$, can be viewed as a sequence of node-disjoint sections. A section of a demand is given by two successive terminals of the demand. The set of sections of demand k can be defined as $\mathcal{T}_k = \{(w_j^k, w_{j+1}^k) : j = 1, \ldots, |T_k|\}$, where $w_{|T_k|+1}^k = w_1^k$. For each demand $k \in K$ and each section $q = (u, v) \in \mathcal{T}_k$, $G^{q,k} = (V^{q,k}, E^{q,k})$ will denote the reduced graph obtained from G by deleting all terminals of the demand k except u and v. Given a section $q = (u, v) \in \mathcal{T}_k$ and a node subset W of $V^{q,k}$ such that $u \in W$ and $v \notin W$, the set of edges of $G^{q,k}$ with one node in W and other in $V^{q,k} \setminus W$ will be called a *section cut* and denoted by $\delta_{G^{q,k}}(W)$.

Figure 3 illustrates this reduction. Figure 3-(a) shows a demand k in G' between (v_1, v_3) routed on paths (v_1, v_3) and (v_1, v_4, v_3). The ordered set T_k is (w_1, w_3, w_4). Considering section $q = (w_4, w_1)$, the corresponding reduced graph given by Figure 3-(b) is obtained by deleting terminal w_3 as well as its incident edges. The set $W = \{w_1, w_2, w_5, w_7\}$ induces the section cut $\delta_{G^{q,k}}(W) = \{e_2, e_6, e_9\}$.

From now on, we will assume that G is complete and each edge e in G has a cost $c(e) > 0$.

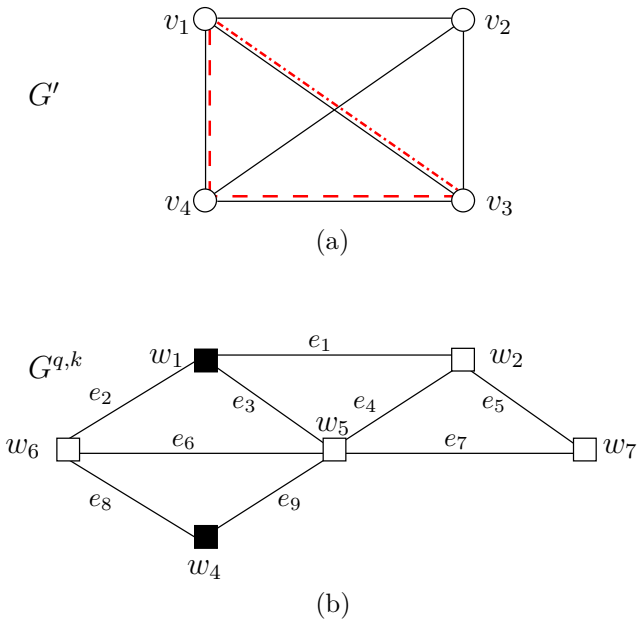

Figure 3: Reduced graph for section $q = (w_4, w_1)$

4. Complexity

The purpose of this section is to show the NP-hardness of the MSOND problem. What we are going to show in what follows is that even the restricted variant of the MSOND problem with a single demand is also NP-hard. The latter will be referred as the SD-MSOND problem. In order to show the NP-hardness of this variant, we shall use the min-sum vertex disjoint paths problem. An instance of the *Min-Sum Vertex Disjoint Paths Problem* (MSVDPP) consists of a graph $G' = (V', E')$, a weight vector $C' = (c'_e,\ e \in E') \geq 0$ associated with the edges of E', a set of pairs of origin-destinations $T' = \{(s_i, t_i), i = 1, ..., k\},\ k \geq 3$, and a positive scalar U. And the problem consists in determining whether or not there exist k node-disjoint paths between the origin-destinations whose total weight does not exceed U. The MSVDPP has been shown to be NP-hard if k is part of the input [16] and it is open for fixed k.

Theorem 1. *The SD-MSOND problem is NP-hard.*

Proof. It is clear that the SD-MSOND problem is in NP. To prove the theorem, we will use a reduction from the MSVDPP. We will show that the MSVDPP reduces to the decision version of the SD-MSOND problem. The latter can be stated as follows: Given a graph $G = (V, E)$, a weight vector $C = (c_e,\ e \in E) \geq 0$ associated with the edges of E, a set of terminals $T = \{(v_1, ..., v_l\} \subseteq V$, and a positive scalar U, the problem consists in determining whether or not there exists in G an elementary cycle going in order through the terminals and whose total weight does not exceed U.

So consider an instance (G',C',T',U) of the MSVDPP where $G' = (V', E')$. We shall construct from (G',C',T',U) an instance (G,C,T,U) of the SD-MSOND problem as follows. We add to G' k vertices $u_1, ..., u_k$ and $2k$ edges $\{t_i, u_i\}$, $\{u_i, s_{i+1}\}, i = 1, ..., k$ ($s_1 = s_{k+1}$). Let $G = (V, E)$ be the resulting graph, and denote by E_u the added edges. Let $c_e = c'_e$ if $e \in E'$ and 0 if $e \in E \setminus E'$ be a weights associated with the edges of G (see Figure 4). Finally, set $T = (s_1, t_1, u_1, s_2, ..., s_j, t_j, u_j..., s_k, t_k, u_k, s_1)$ as the ordered set of terminals.

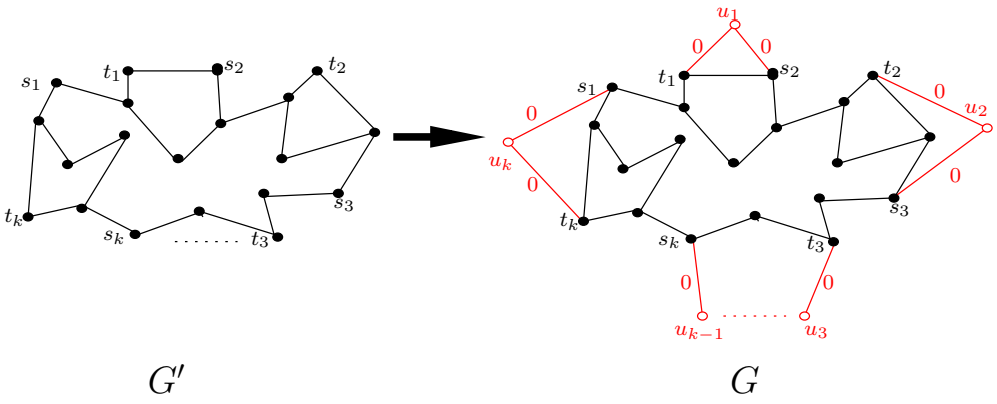

Figure 4: Reduction of the WMSVDPP to the SD-MSOND problem

In the following we show that there exists k vertex disjoint paths between the pairs of T' in G' with total weight $\leq U$ if and only if there exists in G an elementary cycle going in order through the terminals of T with total weight $\leq U$.

Consider first a solution of the MSVDPP in G' with a total weight $c \leq U$. The solution consists of k vertex disjoint paths between the pairs $(s_i, t_i), i = 1, ..., k$. These, together with the set of edges of E_u, constitute an elementary cycle in G going in order through the terminals of T. As the weights of the edges in E_u are equal to 0, the weight of the cycle is equal to $c \leq U$.

Consider now an elementary cycle in G going in order through the terminals T with a total weight $c \leq U$ and consider the sections between the terminals $(s_i, t_i), i = 1, ..., k$. Since the cycle is elementary, these sections are node-disjoint. Moreover, as the weights of all edges in E_u are 0, the total weight of the sub-paths between $(s_i, t_i), i = 1, ..., k$ is exactly equal to c which is $\leq U$.

Therefore, since the number of terminals is not fixed, the SD-MSOND problem is NP-hard. □

5. Integer programming formulations

In this section, integer programming formulations are proposed for the MSOND problem. First we give a cut formulation.

5.1. Cut formulation

Let $x_e^{q,k}$, $e \in E$, $k \in K$, $q \in \mathcal{T}_k$ be a binary variable that takes 1 if edge e is used in section q of demand k and 0 otherwise. These x variables are called *routing variables*. Let y_e, $e \in E$ be a variable which takes 1 if e is installed and 0 otherwise. The y variables are called *design variables*. The cut formulation

for the MSOND problem is given by the following integer linear program,

$$min \sum_{e \in E} c(e) y_e$$

$$\sum_{e \in \delta_{G^{q,k}}(W)} x_e^{q,k} \geq 1 \qquad \begin{array}{l} \text{for all } k \in K,\ q = (u,v) \in \mathcal{T}_k \text{ and} \\ W \subseteq V^{q,k} \text{ st } u \in W \text{ and } v \in \overline{W}, \end{array} \tag{1}$$

$$\sum_{q \in \mathcal{T}_k} \sum_{e \in \delta_G(w)} x_e^{q,k} \leq 2 \quad \text{for all } w \in V,\ k \in K, \tag{2}$$

$$\sum_{q \in \mathcal{T}_k} x_e^{q,k} \leq y_e \qquad \text{for all } e \in E,\ k \in K, \tag{3}$$

$$0 \leq x_e^{q,k} \leq 1 \qquad \text{for all } e \in E,\ k \in K,\ q \in \mathcal{T}_k, \tag{4}$$

$$0 \leq y_e \leq 1 \qquad \text{for all } e \in E, \tag{5}$$

$$x_e^{q,k} \in \{0,1\} \qquad \text{for all } e \in E,\ k \in K,\ q \in \mathcal{T}_k, \tag{6}$$

$$y_e \in \{0,1\} \qquad \text{for all } e \in E. \tag{7}$$

Inequalities (1) are called *section cut inequalities*. They ensure that between the terminals of each section of demand k, there is a path not going through any other terminal of k. Inequalities (2), called *node-disjunction inequalities*, ensure that these paths are node-disjoint, and thus they form an elementary cycle. Inequalities (3) are the *linking inequalities*. These express the fact that an edge which is not installed can not be used to route any section. Constraints (4) and (5) are the *trivial inequalities* and constraints (6) and (7) are the *integrality constraints*.

5.2. Path formulation

In this section we propose a Dantzig-Wolfe decomposition of the integer linear program given by (1)- (7). Let $P^{q,k}$ be the set of paths that can route section $q \in \mathcal{T}_k$ of demand $k \in K$. Recall that these paths are computed in the reduced graph $G^{q,k}$. Let $z_p^{q,k}$, $k \in K$, $q \in \mathcal{T}_k$, $p \in P^{q,k}$, be a binary variable that takes 1 if $p \in P^{q,k}$ is selected to route section q of demand k and 0 otherwise. The z variables are called *path variables* and are linked to the routing variables

by the following relation

$$x_e^{q,k} = \sum_{p \in P^{q,k}} b_p(e) z_p^{q,k}, \tag{8}$$

where $b_p(e)$ takes 1 if edge $e \in p$ and 0 otherwise. Moreover, $a_p(w)$ is defined as

$$a_p^{q,k}(w) = \begin{cases} 1 & \text{if vertex } w \text{ is one of the extremities of path } p, \\ 2 & \text{if } w \text{ is an interior vertex of } p, \\ 0 & \text{otherwise.} \end{cases}$$

Using these notations, we have the following integer programming formulation for the MSOND problem,

$$min \sum_{e \in E} c(e) y_e$$

$$\sum_{p \in P^{q,k}} z_p^{q,k} \geq 1 \qquad \text{for all } k \in K,\ q \in \mathcal{T}_k, \qquad \pi^{q,k} \tag{9}$$

$$\sum_{q \in \mathcal{T}_k} \sum_{p \in P^{q,k}} a_p(w) z_p^{q,k} \leq 2 \qquad \text{for all } w \in V,\ k \in K, \qquad \lambda_w^k \tag{10}$$

$$\sum_{q \in \mathcal{T}_k} \sum_{p \in P^{q,k}} b_p(e) z_p^{q,k} \leq y_e \qquad \text{for all } e \in E,\ k \in K, \qquad \beta_e^k \tag{11}$$

$$0 \leq y_e \leq 1,\ y_e \in \{0,1\} \qquad \text{for all } e \in E, \tag{12}$$

$$0 \leq z_p^{q,k} \leq 1,\ z_p^{q,k} \in \{0,1\} \qquad \text{for all } k \in K,\ q \in \mathcal{T}_k,\ p \in P^{q,k}. \tag{13}$$

The above formulation will be called *path formulation*, and denoted by (P). Note that (P) contains an exponential number of z variables. In the following we devise a Branch-and-Price algorithm over (P).

6. Branch-and-Price Algorithm

As (P) has a huge number of variables, solving its linear relaxation requires a column generation method. The combination of column generation with the Branch-and-Bound technique yields a Branch-and-Price algorithm for solving (P). The idea of column generation is to solve a sequence of linear programs containing a restricted number of z variables (called also columns), defining the

so-called *Restricted Linear Programs* (RLP). Starting from an initial linear program (RLP_{ini}) with a reasonable number of variables, at each iteration of the algorithm, we solve a satellite problem called the *pricing problem*. The pricing aims at detecting additional columns with negative reduced cost. If no such column exists, the current (RLP) is optimal. However, if its solution is fractional, a branching should be performed. Algorithm 1 summarizes the steps of the Branch-and-Price algorithm.

Algorithm 1: Branch-And-Price Algorithm

Data: A MSOND instance

Result: Optimal solution for (P)

```
RLP ← RLP_ini;
repeat
    newColumn ← false;
    solve the linear program RLP and denote by (z*, y*) and (π*, λ*, β*)
    its optimal primal and dual solutions;
    forall the k ∈ K, q ∈ T_k do
        solve the associated pricing problem using (π*, λ*, β*);
        if the optimal value of the pricing problem is negative then
            add to RLP the z variable corresponding to the pricing
            optimal solution;
            newColumn ← true;
until newColumn=false;
if (z*, y*) is integer then
    return (z*, y*);
else
    apply the primal heuristic described in Algorithm 2;
    create two subproblems using a branching rule and solve those
    subproblems recursively;
    return the best subproblem solution.
```

In the following sections, we describe in more details the main steps of Algorithm 1.

6.1. Initial linear program RLP_{ini}

For each section $q = (u, v)$ of a demand k, let $P_0^{q,k}$ be the subset of $P^{q,k}$ given by the paths between u and v, of the form (u, v), (u, s_1, v) and (u, s_1, s_2, v), where $s, s_1, s_2 \in S_k$. Our first linear program RLP_{ini} is given by the columns corresponding to the paths of $\bigcup_{q,k} P_0^{q,k}$. Since the $|T_k|$ direct paths of each demand k form an elementary cycle going in order through the terminals in T_k, it is clear that RLP_{ini} is feasible. The purpose of also having the paths with one or two Steiner vertices in RLP_{ini} is accelerating the Branch-and-Price, reducing the number of columns that have to be generated along the algorithm. Experiments have shown that, for most of the instances, it is not worthy to include paths with three or more Steiner vertices in the initialization, since they make the RLP solution too slow.

6.2. Pricing problem

The Branch-and-Price algorithm works over a restricted linear program involving only a subset of all z variables. Additional columns will then be generated when they are needed. This can be achieved by solving the so-called pricing problem.

For every demand $k \in K$ and section $q \in \mathcal{T}_k$, we consider a pricing problem which consists in finding a new variable $z_p^{q,k}$ with negative reduced cost. Let $\pi^{q,k}$, λ_w^k and β_e^k be the dual variables associated with inequalities (9), (10) and (11), respectively. The reduced cost for variable $z_p^{q,k}$ is then given by

$$R_p^{q,k} = -(\pi^{q,k} + \sum_{w \in V^{q,k}} a_p(w)\lambda_w^k + \sum_{e \in E^{q,k}} b_p(e)\beta_e^k).$$

Therefore, given an optimal dual solution (π, λ, β) of RLP, the pricing problem consists in finding a path p^* of $P^{q,k}$ such that

$$R_{p^*}^{q,k} = \min_{p \in P_k^q} R_p^{q,k}. \qquad \text{(PS}(q,k)\text{)}$$

If $R_{p^\star}^{q,k} < 0$, then the column related to variable $z_{p^*}^{q,k}$ is added to RLP.

In what follows, we will show that the pricing problem PS(q,k) reduces to a shortest path problem in the reduced graph $G^{q,k}$. Suppose $q = (w_j, w_{j+1})$. Associate with each edge $e = uv$ of graph $G^{q,k}$ the length

$$l_e = \begin{cases} -\beta_e^k & \text{if } uv = w_j w_{j+1}, \\ -\beta_e^k - \lambda_v^k & \text{if } u \in \{w_j, w_{j+1}\},\ v \in V^{q,k} \setminus \{w_j, w_{j+1}\}, \\ -\beta_e^k - \lambda_u^k & \text{if } v \in \{w_j, w_{j+1}\},\ u \in V^{q,k} \setminus \{w_j, w_{j+1}\}, \\ -\beta_e^k - \lambda_u^k - \lambda_v^k & \text{if } u, v \in V^{q,k} \setminus \{w_j, w_{j+1}\}. \end{cases}$$

See Figure 5 for an illustration that considers the reduced graph of Figure 3 corresponding to section $q = (w_4, w_1)$.

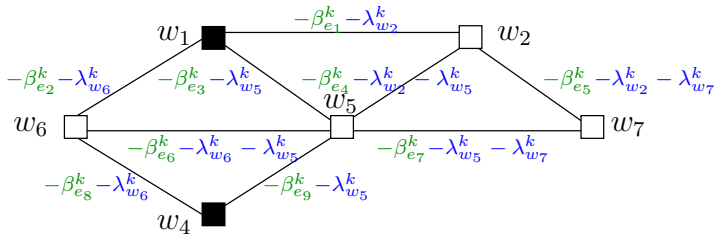

Figure 5: Reduced graph with dual weights on edges

345 **Lemma 2.** *Given a path $p \in P^{q,k}$, $R_p^{q,k} = -\pi^{q,k} - \lambda_{w_j}^k - \lambda_{w_{j+1}}^k + l(p)$, where $l(p)$ is the length of p.*

Proof. Assume p is the sequence of nodes and edges $(w_j, e_1, u_1, ..., u_{t-1}, e_t, w_{j+1})$.

$$\begin{aligned} R_p^{q,k} &= -\pi^{q,k} - \sum_{w \in V^{q,k}} a_p^{q,k}(w)\lambda_w^k - \sum_{e \in E^{q,k}} b_p^{q,k}(e)\beta_e^k \\ &= -\pi_t^{q,k} - \lambda_{w_j}^k - \lambda_{w_{j+1}}^k - 2\sum_{i=1}^{t-1} \lambda_{u_i}^k - \sum_{i=1}^{t} \beta_{e_i}^k \\ &= -\pi_t^{q,k} - \lambda_{w_j}^k - \lambda_{w_{j+1}}^k + l(p). \end{aligned}$$

□

By Lemma 2, minimizing $R_p^{q,k}$ reduces to minimizing $l(p)$, that is to say solving a shortest path problem in the reduced graph $G^{q,k}$. Note that dual

variables λ and β are non-positive. Hence, edge lengths are nonnegative and the shortest path problem can be solved in polynomial time using for example Dijkstra Algorithm [17].

6.3. Branching scheme

At each node of the Branch-and-Price tree, if the optimal solution (z^*, y^*) of the associated RLP is not integer, a branching must be performed. If possible, this is done over the design variables. If there are y variables with fractional value, the most fractional such variable is selected, say y_e^*. On the first subproblem y_e^* is fixed to 0, on the second subproblem y_e^* is fixed to 1. Notice that this kind of branching does not change the pricing.

However, at a certain level of the tree, we may have a fractional solution where all the design variables are integral. In this case, a branching on the path variables is necessary. The challenge is to identify a branching rule that eliminates the current fractional solution without compromising the tractability of the pricing problem. In general, the branching rules for path-based formulations are defined on the original edges (or arcs) of the paths [18, 19]. For example, we can branch as follows: for a given $k \in K$ and $q \in \mathcal{T}_k$, on the first subproblem we forbid the use of a certain edge e on the path that connects that section; on the second subproblem we impose that it passes by e. This amounts on branching on the following constraints:

- $\sum_{p \in P^{q,k}} b_p^{q,k}(e) z_p^{q,k} \leq 0$ for the first subproblem,
- and $\sum_{p \in P^{q,k}} b_p^{q,k}(e) z_p^{q,k} \geq 1$ for the second subproblem.

The dual variables of these constraints change the reduced cost calculation when solving PS(q, k). For the first constraint this poses no problem, since the new $l(e)$ will still be non-negative. However, the dual variable of the second constraint may be positive, resulting in a negative edge length $l(e)$. This may yield negative cycles in the graph. Hence, the shortest path problems in the pricing can no longer be solved with a polynomial algorithm. We then decided to mitigate this difficulty using the following strategy.

As all the design variables y_e^* are integer, we choose a fractional path variable, say $z_p^{*q,k}$. Assume that this variable is such that $q = (w_0, w_1)$ and $p = (w_0, s_1, s_2, ..., s_j, w_1)$. The idea is to choose in a clever way the edge for branching. For this end, we opt for the first edge of the selected path p, namely $e_0 = (w_0, s_1)$. On the first branch, the use of e_0 in paths of $P^{q,k}$ is forbidden. This can be done by deleting the edge e_0 when solving the corresponding shortest path problem. On the second branch, e_0 is imposed in the paths of $P^{q,k}$. Since the first extremity of e_0 coincides with w_0, the pricing problem reduces to a shortest path calculated in the reduced graph $G^{q,k}$ between s_1 and w_1. At a given depth of the Branch-and-Price tree, we may have to branch again on paths of section q and demand k. In this case, $e_1 = (s_1, s_2)$ will be the preferred edge of branching, for which the process described above is similarly applied. By doing this, for each q and k, a sequence of edges forming a subchain originating in w_0, may be imposed without changing the pricing. Note that it is also possible, in a similar way, to choose e_0 (e_1, etc...) as the last edge of path p. In this case, for each q and k, we may impose a chain of edges starting at w_0 and another chain of edges ending in w_1 without changing the pricing.

Consider again the example of Figure 5. Suppose that for section $q = (w_1, w_4)$, there exists a variable $z_p^{*q,k}$ with a fractional value associated with the path $p = \{w_1, w_6, w_5, w_4\}$. Based on this variable, two subproblems are created:

- in the first subproblem, the use of edge (w_1, w_6) is forbidden, and the pricing problem is a shortest path from w_1 to w_4, calculated in the graph represented in the left hand side of Figure 6.
- in the second subproblem, we impose the use of edge (w_1, w_6). In this case, the pricing problem reduces to a shortest path from w_6 to w_4 in the graph represented in the right hand side of Figure 6.

Note that this branching strategy is not complete. Theoretically, we may have a fractional solution where, for every k and q, any edge e_0 that joins the current chains of fixed edges satisfies $\sum_{p \in P^{q,k}} b_p^{q,k}(e_0) z_p^{q,k} = 1$, which means

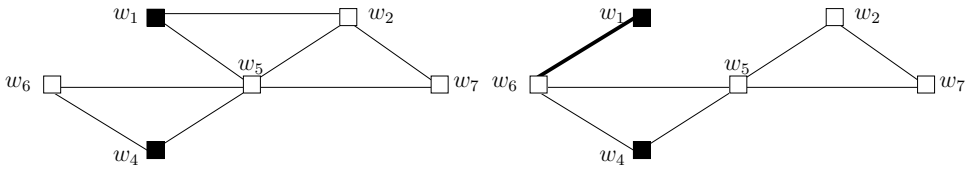

Figure 6: Illustration of the branching strategy

that these edges are not fractional. In that case, it may be necessary to branch on edges in the so-called *divergence point* defined in [20], increasing the complexity of the pricing. This unlikely case never happened in our experiments and consequently the procedure was applied as it has been described above.

To accelerate the Branch-and-Price algorithm, we propose a primal heuristic that enables the improvement of the upper bound and a faster pruning of the Branch-and-Bound tree.

6.4. Primal heuristic

Given a fractional solution, a feasible integer solution for the MSOND problem can be determined using the primal heuristic described in Algorithm 2. This is a greedy-like algorithm. We start from a feasible fractional solution (z, y). For each demand $k \in K$, we go through the different sections from (w_1, w_2) to $(w_{|\mathcal{T}_k|}, w_{|\mathcal{T}_k|+1})$ with $w_{|\mathcal{T}_k|+1} = w_1$. At step i of Algorithm 2, our objective is to select a path for routing section $q = (w_i, w_{i+1})$ among the paths corresponding to the nonzero variables $z_p^{q,k}$. This path must respect the disjunction constraints with paths already chosen for routing the previous sections.

In Algorithm 2, $S_{k,+}$ represents the Steiner nodes that have been used to route the $(i-1)$ first sections of demand k. Considering section $q = (w_i, w_{i+1})$, we first begin by sorting in decreasing order the variables $z_p^{q,k}$ by their values. Let $z_{p_1}^{q,k} \geq z_{p_2}^{q,k} \geq ... \geq z_{p_h}^{q,k}$ be the sorted variables representing respectively paths $p_1, p_2, ..., p_h$ routing $q \in \mathcal{T}_k$. The first path p_j such that its set of Steiner nodes Sp_j is disjoint from $S_{k,+}$ is selected ($\hat{z}_{p_j}^{q,k} = 1$) and Sp_j is added to $S_{k,+}$. If none of the paths $p_1, p_2, ..., p_h$ has been selected, then the section is routed by the path $p_0 = (w_i, w_{i+1})$ ($\hat{z}_{p_0}^{q,k} = 1$). Once a path is selected to route section

$q = (w_i, w_{i+1})$, the next section is considered and the same procedure is applied.

Algorithm 2: Primal Heuristic

Data: Fractional Solution (z, y)

Result: Integer Feasible Solution $(\hat{z}, \hat{y})$

$(\hat{z}, \hat{y}) \leftarrow (0, 0)$;
forall the $k \in K$ **do**
  $S_{k,+} = \emptyset$;
  for $i \in \{1, ..., |\mathcal{T}_k|\}$ **do**
    $q \leftarrow (w_i, w_{i+1})$;
    sort $z_p^{q,k}$ by decreasing order;
    Let $p_1, p_2, ..., p_h$ be such that $z_{p_1}^{q,k} \geq z_{p_2}^{q,k} \geq ... \geq z_{p_h}^{q,k}$;
    for $j = 1$ *to* h **do**
      if $Sp_j \cap S_{k,+} = \emptyset$ **then**
        $\hat{z}_{p_j}^{q,k} \leftarrow 1$;
        $S_{k,+} = S_{k,+} \cup Sp_j$;
        $j \leftarrow h + 1$;
        break;
    if $\sum_{j=1}^{h} \hat{z}_{p_j}^{q,k} = 0$ **then**
      `/* route the section` q `on the edge` (w_i, w_{i+1}) `*/`
      denote $p_0 = (w_i, w_{i+1})$;
      $\hat{z}_{p_0}^{q,k} \leftarrow 1$;
update design variables $\hat{y}$;
return the integer feasible solution $(\hat{z}, \hat{y})$;

The primal heuristic described in Algorithm 2 is applied at the end of the column generation phase for each node of the Branch-and-Price tree.

7. Computational results

The Branch-and-Price algorithm described in the previous section has been implemented in C++, using ABACUS (A Branch-And-Cut System) 3.2 [21] to

manage the Branch-and-Price tree and Cplex 12.0 as LP-solver [22]. The tests were performed on a Bi-Xeon quad-core E5507 2.27GHz machine with 8Go of RAM, running under Linux. The maximum CPU time was fixed to 3 hours.

Two types of instances were used:

- Random instances with Euclidean edge costs. For each value of $|V|$ and $|K|$, a group of 5 instances were generated, taking the coordinates of the first $|V|$ vertices in TSP-Library instances [23] $a280$, $bier127$, $eil101$, $lin105$ and $tsp225$, respectively. The first $|V'|$ vertices of an instance are their terminals (we choose $|V'| = |V| - 2$). The $|K|$ demands are generated as follows. First the cardinality of the set T_k corresponding to a demand $k \in K$ is randomly chosen between 3 and 6. Then the vertices in T_k are are randomly chosen among the terminals.

- Realistic instances adapted from the SNDLib instances $dfn-bwin$, $polska$, $nobel-us$, $newyork$ and $geant$. For these instances, V is the vertex-set of their original SNDLib instance, V' is taken as equal to V. The $|K|$ original demands with larger demand value define the origin O_k and destination D_k of each demand $k \in K$. The direct edges (O_k, D_k) are removed, as well as some other edges of the complete graph, and for each k, two node-disjoint shortest paths between O_k and D_k are computed. The remaining terminals of demand k are the vertices visited in those paths.

In the sequel, the entries of the different tables are :

$\|V\|$	:	number of nodes in the optical layer (graph G);
$\|K\|$	:	number of demands;
Terminals	:	average number of terminals by demand;
Path-ini	:	number of columns provided by the initial solution;
Path-gen	:	number of columns generated during the pricing;
Nodes	:	number of nodes in the tree;
Gap	:	the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root node of the Branch-and-Price tree;
Opt	:	number of instances in the group solved to optimality; (for the random instances only);
CPU	:	total CPU time, given in h:mm:ss for realistic instances and in seconds for the random instances, the later are averages over the solved instances of the group.

Tables 1 and 2 report the results for the random instances. All instances with $|V| \leq 14$, with $|K|$ up to 16, could be solved to optimality with the time limit. However, larger values of $|V|$ make the problem much more difficult to solve. When $|V| = 25$, only instances with few demands could be solved to optimality. Some observations:

- The number of paths generated during the pricing procedure (Path-gen) is not very large for the smaller instances. This is thanks to the good quality of the initial solution given by Path-ini. In fact, for 23 groups of instances, the optimum was reached reached with less than 100 generated columns in average.

- For 44 groups where all instances were solved, the root lower bounds were situated at most at 10% from the optimum value. In those cases, the time necessary to obtain these bounds did not exceed some seconds. However, for the larger and harder groups of instances, some large gaps were observed, reaching over 40% for the instances with $|V| = 25$ and

$|K| \geq 5$.

Tables 3 and 4 report results of the Branch-and-price algorithm for the realistic instances. Results are given for 70 instances with a number of nodes ranging from 10 to 22 and a number of demands going from 2 to 30. It can be seen that 59 instances could be solved to optimality within the time limit, 49 of them within 10 minutes. In particular, all the instances of *dfn-bwin* with 10 nodes and demands from 2 to 30 have been solved to optimality in less than 2 minutes. Usually, those more realistic instances are easier than the previous random instances with the same size. For example, the *geant* instances with $|V| = 22$ could be solved with $|K|$ up to 12. This behavior is somehow expected. The random instances may have some demands where the terminals are very far apart, making the solution harder. This does not happen in the realistic instances. For the two last instances of *geant*, it was not possible to solve the linear relaxation within the time limit, so a " $-$ " appears in the Gap column.

The realistic instances in a group, corresponding an original SNDLib instance, are *nested*, in the sense that an instance with $|K| = 4$ is the instance with $|K| = 2$ with 2 additional demands, and so on. It is expected that the instances in a group became more difficult as the number of demands increase. However, a peculiar behavior was observed in the group *newyork*. When $|K|$ was increased from 6 to 8, the resulting instance suddenly became much harder and could not be solved within the time limit. This difficulty persists also for the instances with 10 and 12 demands. However, by adding more demands, the resulting instances became easier, and could be solved to optimality with up to $|K| = 30$.

After an investigation, it was discovered that the increase from 6 to 8 produced a demand i where $T_i = \{13, 6, 14, 0\}$ and another demand j with $T_i = \{14, 13, 6, 0\}$ (see the illustration in Figure 7). This situation, where two demands share a number of terminals with conflicting order constraints increase the root gaps and make the instance much harder. However, when $|K|$ was increased from 12 to 14, a third demand $\{0, 6, 8, 7\}$ was added, the gap decreased

$\|V\|$	$\|K\|$	Terminals	Path-ini	Path-gen	Nodes	Gap	Opt	CPU-time
6	2	3	60	0.2	1	0	5/5	0.6
6	4	3	120	0.6	1	0	5/5	0.6
6	5	3	150	1	1	0	5/5	0.6
6	6	3	180	0.6	1	0	5/5	0.6
8	2	4.5	118	0.4	1	0	5/5	0.6
8	4	4.5	236	2.6	8.6	7.33	5/5	0.5
8	5	4.2	314	5.4	9.8	7.33	5/5	0.3
8	6	4.17	382	6.6	12.6	5.01	5/5	0.2
8	8	4.13	518	6.8	7.4	0.39	5/5	0.3
8	10	4.1	646	5.2	9	2.14	5/5	0.4
8	12	4.08	782	3.8	7	2.14	5/5	0.4
8	15	4.07	986	4.6	13	4.35	5/5	0
10	2	4.5	278	1.8	1	0	5/5	0.6
10	4	4.25	558	78.6	10.2	3.23	5/5	3.3
10	5	4.6	660	60.2	17	1.14	5/5	3
10	6	4.67	790	78.2	24.6	1.26	5/5	7.5
10	8	4.75	1010	20.2	15	2.11	5/5	0.2
10	10	5.1	1182	15.8	25.4	5.32	5/5	0
10	12	4.92	1478	20.2	28.6	5.32	5/5	0
10	15	4.93	1858	50.2	75	8.37	5/5	15.4
10	16	5.5	1960	47.2	63.8	8.57	5/5	19.2
12	2	6.5	404	16.8	17.8	3.6	5/5	0.3
12	4	5.5	914	272.2	45.4	3.91	5/5	21.9
12	5	5.8	1096	313	112.6	7.24	5/5	40.2
12	6	5.5	1356	453.2	101.4	7.4	5/5	50.9
12	8	5.63	1800	366.4	202.6	6.23	5/5	74.4
12	10	5.7	2232	269.2	127.4	7.97	5/5	54.4
12	12	5.92	2590	288.2	302.6	7.37	5/5	120.3
12	15	6	3204	536.8	935	8.86	5/5	595.8
12	16	6	3426	582	1201.8	8.86	5/5	837.3
14	2	4.5	814	46	1	0	5/5	0
14	4	4	1584	327.6	1.8	0	5/5	4
14	5	4.4	1974	7585.8	480.2	5.08	5/5	2042.8
14	6	4.5	2384	4676.6	235	3.59	5/5	903.1
14	8	4.88	3084	3839.4	303.8	7.1	5/5	604.6
14	10	4.9	3878	5405	545.8	6.4	5/5	1371.9
14	12	4.92	4698	6009.4	887.4	6.69	5/5	2643.9
14	15	5	5770	3362.6	871	8.08	5/5	1954.9
14	16	4.94	6174	3431	886.6	8.08	5/5	2206.4

Table 1: Branch-and-Price results for random instances (1)

$\|V\|$	$\|K\|$	Terminals	Path-ini	Path-gen	Nodes	Gap	Opt	CPU-time
16	2	4.5	1190	74.8	1	0	5/5	0
16	4	4.5	2306	847.2	7	0.39	5/5	28.8
16	5	4.8	2912	9263	450.2	3.87	5/5	2334.2
16	6	4.83	3522	11858.6	533	6.3	5/5	3403.1
16	8	4.88	4708	14177.4	764.6	7.11	2/5	7156.1
16	10	5.4	5802	11848.2	1174.2	11.08	2/5	7959.9
16	12	5.58	6982	12233	1306	24.17	0/5	10800
16	15	5.93	8628	7406.2	1211.2	29.87	0/5	10800
16	16	5.75	9138	6993	1050.8	26.48	0/5	10800
18	2	5	1658	125	1.4	0	5/5	0
18	4	4.75	3206	2642.4	14.6	1.29	5/5	221.6
18	5	5.2	4060	14849	327.6	5.97	3/5	6533.9
18	6	4.83	4738	14295	255	6.01	2/5	6564.3
18	8	5	6224	17655.4	382.4	13.62	1/5	8829.9
18	10	5.3	7948	17216.4	529.6	21.43	0/5	10800
18	12	5.5	9672	13843.4	558.6	25.29	0/5	10800
18	15	5.86	12008	11597.8	369.2	32.63	0/5	10800
18	16	5.5	12878	11041.2	316.2	32.06	0/5	10800
20	2	4.5	2158	166.6	1	0	5/5	0
20	4	4.25	4158	2789.8	9.4	0.01	5/5	246.3
20	5	4.6	5340	18493.4	156.4	2.89	2/5	7549.8
20	6	4.83	6522	21695.2	126.4	7.3	1/5	9429.5
20	8	5.13	8710	23298.2	101.8	13.18	0/5	10800
20	10	5.6	11060	19414.8	257.2	30.21	0/5	10800
20	12	5.75	13432	16025.2	175.4	34.01	0/5	10800
20	15	6.07	16874	14645.2	123.6	46.8	0/5	10800
20	16	5.63	18004	14225	106.6	46.7	0/5	10800
25	2	4.5	3627	241.6	1	0	5/5	5.5
25	4	4.75	7357	7672.6	52.2	0.53	4/5	3257.7
25	5	5.4	9677	2162.8	0.8	46.72	0/5	10800
25	6	5.33	11682	4758.6	2.2	50.98	0/5	10800
25	8	5.25	15622	5858	3	69.65	0/5	10800
25	10	5.4	19902	6468	2.6	59.25	0/5	10800
25	12	5.42	24079	7702.4	2.2	62.10	0/5	10800
25	14	5.43	28256	7608	1.8	48.71	0/5	10800
25	16	5.38	32292	2644.4	1	-	0/5	10800

Table 2: Branch-and-Price results for random instances (2)

and the resulting instances could be solved.

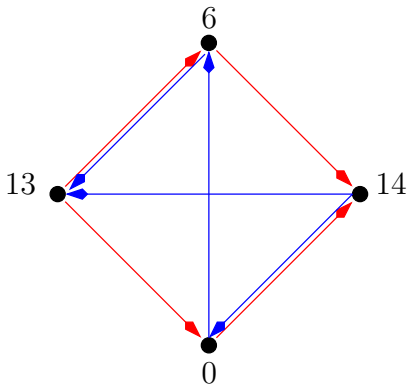

Figure 7: Order conflicts between demands

Figure 8 shows an optimal solution for a realistic instance with 20 nodes representing French cities, there are 15 demands. This instance has been generated as follows. The coordinates of the cities are given by their longitude and latitude, respectively. Demands are randomly generated so as to have a non-trivial instance. Figure 8 shows that, for example, the demand (Paris, Strasbourg) whose logical paths are (Paris, Lille, Strasbourg) and (Paris, Dijon, Strasbourg), is routed by the physical paths (Paris, Reims, Lille, Strasbourg) and (Paris, Dijon, Strasbourg).

Our current model does not consider the length of the paths. However, in practice, routing using long paths in optical networks can imply additional costs necessary to regenerate the optical wavelengths whose signals become weaker. In the solution shown in Figure 8, the demand between (Marseille, Rennes) whose logical paths are (Marseille, Le-Mans, Rennes) and (Marseille, Reims, Rennes), is routed by the physical paths (Marseille, Montpellier, Limoges, Bordeaux, Poitiers, Le-Mans, Nantes, Brest, Rennes) and (Marseille, Lyon, Dijon, Reims, Paris, Rennes). This means that the section (Marseille, Le Mans) is routed through a path a length 5. Adding constraints that limit the paths' lengths for the MSOND problem, known as *the hop constraints*, could be an interesting

Instance	$\|V\|$	$\|K\|$	Terminals	Path-ini	Path-gen	Nodes	Gap	CPU-time
dfn-bwin	10	2	3.5	298	4	1	0	0:00:00
dfn-bwin	10	4	3.25	598	2	1	0	0:00:00
dfn-bwin	10	5	3.2	748	12	1	0	0:00:01
dfn-bwin	10	6	3.29	898	48	1	0	0:00:01
dfn-bwin	10	8	3.33	1176	65	1	0	0:00:03
dfn-bwin	10	10	3.31	1476	155	1	0	0:00:05
dfn-bwin	10	12	3.39	1776	523	1	0	0:00:10
dfn-bwin	10	14	3.5	2076	615	1	0	0:00:15
dfn-bwin	10	15	3.45	2224	483	1	0	0:00:11
dfn-bwin	10	16	3.48	2374	542	1	0	0:00:11
dfn-bwin	10	18	3.5	2670	1086	5	2.13	0:00:50
dfn-bwin	10	20	3.25	2966	971	23	5.39	0:01:12
dfn-bwin	10	25	3.53	3710	1201	63	6.31	0:02:15
dfn-bwin	10	30	3.56	4454	916	37	6.14	0:01:50
polska	12	2	3	492	56	1	0	0:00:02
polska	12	4	3.75	1002	155	1	0	0:00:05
polska	12	5	3.8	1262	573	7	2.96	0:00:23
polska	12	6	3.71	1508	580	3	1.68	0:00:14
polska	12	8	3.67	1976	2259	63	4.5	0:02:33
polska	12	10	3.69	2472	2358	61	4.5	0:02:32
polska	12	12	3.72	2978	3174	51	5.92	0:03:49
polska	12	14	3	3470	4130	51	5.92	0:06:23
polska	12	15	3.65	3716	4668	55	5.92	0:08:06
polska	12	16	3.56	3976	5323	47	4.66	0:08:34
polska	12	18	3.53	4496	3092	17	3.46	0:03:33
polska	12	20	3.75	4988	4334	21	3.46	0:06:48
polska	12	25	3.53	6232	4384	21	3.46	0:09:48
polska	12	30	3.53	7490	7205	55	5.71	0:32:02
nobel-us	14	2	3.5	770	25	1	0	0:00:01
nobel-us	14	4	4	1584	102	1	0	0:00:02
nobel-us	14	5	3.8	1950	58	1	0	0:00:02
nobel-us	14	6	3.64	2316	124	1	0	0:00:04
nobel-us	14	8	3.6	3048	379	1	0	0:00:09
nobel-us	14	10	3.63	3780	812	1	0	0:00:24
nobel-us	14	12	3.67	4588	1044	1	0	0:00:32

Table 3: Branch-and-Price results for realistic instances (1)

Instance	$\|V\|$	$\|K\|$	Terminals	Path-ini	Path-gen	Nodes	Gap	CPU-time
nobel-us	14	14	3.5	5344	824	11	4.18	0:00:55
nobel-us	14	15	3.75	5710	590	11	4.18	0:00:44
nobel-us	14	16	3.76	6114	1172	19	5.18	0:01:47
nobel-us	14	18	3.7	6922	1948	17	2.62	0:02:55
nobel-us	14	20	4	7736	2294	17	1.91	0:03:31
nobel-us	14	25	3.63	9718	1579	1	0	0:01:43
nobel-us	14	30	3.64	11624	19299	69	3.44	3:00:00
newyork	16	2	3	1020	128	1	0	0:00:06
newyork	16	4	3	2040	53	1	0	0:00:03
newyork	16	5	3	2550	139	1	0	0:00:06
newyork	16	6	3.36	3130	983	1	0	0:00:25
newyork	16	8	3.4	4220	24366	263	8.02	3:00:00
newyork	16	10	3.38	5240	23602	211	7.57	3:00:00
newyork	16	12	3.44	6260	22721	143	7.35	3:00:00
newyork	16	14	3	7450	14599	83	4.94	1:12:02
newyork	16	15	3.5	8030	16625	97	5.03	1:47:26
newyork	16	16	3.56	8540	18103	131	5.03	2:05:26
newyork	16	18	3.63	9660	6919	31	1.35	0:17:42
newyork	16	20	3	10820	10277	27	4.05	0:37:41
newyork	16	25	3.6	13650	8849	31	2.97	0:31:00
newyork	16	30	3.56	16436	7688	51	3.28	0:46:13
geant	22	2	3.5	2386	33	1	0	0:00:01
geant	22	4	3.75	4922	186	1	0	0:00:06
geant	22	5	3.67	6008	835	1	0	0:00:23
geant	22	6	3.67	7308	1120	1	0	0:01:09
geant	22	8	3.5	9480	4932	1	0	0:10:44
geant	22	10	3.4	11652	9319	1	0	0:30:13
geant	22	12	3.33	13824	11491	1	0	1:46:31
geant	22	14	3.43	16424	16133	5	11.38	3:00:00
geant	22	15	3.56	17724	17207	3	10.16	3:00:00
geant	22	16	3.5	19024	17861	3	7.01	3:00:00
geant	22	18	3.56	21560	28849	3	6.67	3:00:00
geant	22	20	3.5	23732	33268	3	6.92	3:00:00
geant	22	25	3.44	29376	14769	1	-	3:00:00
geant	22	30	3.53	35748	49111	1	-	3:00:00

Table 4: Branch-and-Price results for realistic instances (2)

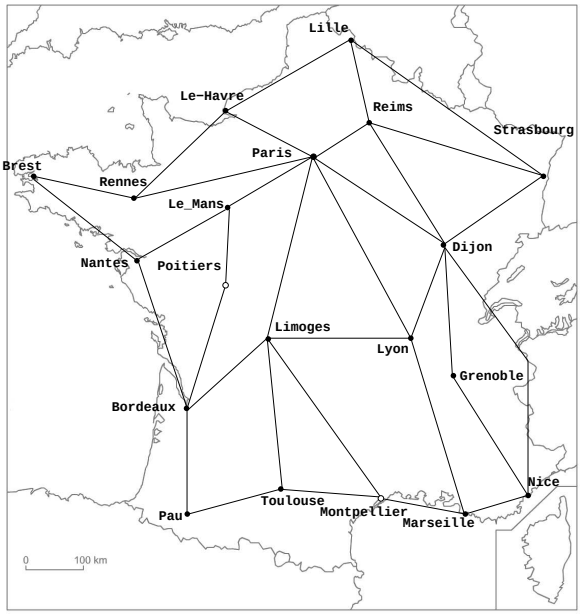

Figure 8: Solution for a French instance with 20 nodes and 15 demands

direction to explore in the future.

8. Concluding remarks

In this paper we have considered a multilayer survivable network design problem with application in the design of reliable IP-over-optical networks. We have shown that it is NP-hard even for one demand. We proposed for the problem an integer programming formulation using path variables and described a Branch-and-Price algorithm to solve it. We discussed in particular the pricing problem, the branching strategy and we proposed an appropriate primal heuristic. Our computational results have shown that the algorithm performs reasonably well for realistic sized instances.

There are many interesting extensions that could be considered in future works. As already mentioned, it would be interesting to have hop constraints on the paths. Another extension would be to consider the capacity dimensioning of the network in addition to the survivability aspect. Finally, an interesting (and

challenging) perspective is to have a global view of the IP-over-optical network and look for models that consider the design of the two layers simultaneously.

References

[1] E. Zouganelli, Toptical network functionality: From "dumb fat pipes" to bright networking, Telektronikk 97 (2001) 346–354.

[2] G. Dahl, A. Martin, M. Stoer, Routing through virtual paths in layered telecommunication networks, Operations Research 47 (5) (1999) pp. 693–702.

[3] S. Borne, E. Gourdin, B. Liau, A. R. Mahjoub, Design of survivable ip-over-optical networks, Annals of Operations Research 146 (1) (2006) 41–73.

[4] P. Belotti, A. Capone, G. Carello, F. Malucelli, Multi-layer mpls network design: The impact of statistical multiplexing, Computer Networks 52 (6) (2008) 1291–1307.

[5] L. Gouveia, P. Patrício, A. de Sousa, Hop-constrained node survivable network design: An application to mpls over wdm, Networks and Spatial Economics 8 (2008) 3–21.

[6] S. Orlowski, C. Raack, A. M. C. A. Koster, G. Baier, T. Engel, P. Belotti, Branch-and-cut techniques for solving realistic two-layer network design problems, in: A. Koster, X. Muoz (Eds.), Graphs and Algorithms in Communication Networks, Texts in Theoretical Computer Science. An EATCS Series, Springer Berlin Heidelberg, 2010, pp. 95–118.

[7] S. Orlowski, M. Pióro, Complexity of column generation in network design with path-based survivability mechanisms, Networks 59 (1) (2012) 132–147.

[8] S. Raghavan, D. Stanojevic, Branch and price for wdm optical networks with no bifurcation of flow, INFORMS Journal on Computing 23 (1) (2011) 56–74.

[9] S. Bradner, The recommendation for the ip next generation protocol, IETF.

[10] T. Jensen, Internet protocol and transport protocols, Telektronikk 97 (2001) 20–38.

[11] B. Rajagopalan, D. Pendarakis, D. Saha, R. S. Ramamoorthy, K. Bala, Ip over optical networks: Architectural aspects, Communications Magazine, IEEE 38 (9) (2000) 94–102.

[12] G. Cornujols, J. Fonlupt, D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra, Mathematical Programming 33 (1985) 1–27.

[13] M. Baïou, A. Mahjoub, Steiner 2-edge connected subgraph polytopes on series-parallel graphs, SIAM J. Discrete Math. 10 (2002) 505–514.

[14] J. Salazar-González, The steiner cycle polytope, European Journal of Operational Research 147 (2003) 671–679.

[15] M. Steinová, Approximability of the minimum steiner cycle problem, Computing And Informatics 29 (6).

[16] Y. Kobayashi, C. Sommer, On shortest disjoint paths in planar graphs, Mathematical Engineering Technical Reports.

[17] E. W. Dijkstra, A note on two problems in connection with graphs, Numerische Mathematik 1 (1959) 269–271.

[18] C. Barnhart, C. A. Hane, P. H. Vance, Using Branch-and-Price-and-Cut to solve origin-destination integer multicommodity Flow problem, Operations Research 48 (2000) 318–326.

[19] D. Feillet, A tutorial on column generation and branch-and-price for vehicle routing problems, J. Oper. Res. 8 (2010) 407–424.

[20] C. Barnhart, C. A. Hane, P. H. Vance, Integer multicommodity flow problems, in: Integer Programming and Combinatorial Optimization, Springer, 1996, pp. 58–71.

[21] http://www.informatik.uni-koeln.de/abacus/.

[22] http://www.ilog.com/products/cplex/.

[23] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.