European Journal of Operational Research 257 (2017) 25–40

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Regenerator location problem: Polyhedral study and effective branch-and-cut algorithms

Xiangyong Li[a,*], Y. P. Aneja[b]

[a] School of Economics & Management, Tongji University, Shanghai 200092, China
[b] Odette School of Business, University of Windsor, Windsor, Ontario N9B 3P4, Canada

ARTICLE INFO

Article history:
Received 30 October 2015
Accepted 16 July 2016
Available online 21 July 2016

Keywords:
Optical network design
Regenerator placement
Integer programming
Branch and cut
Facets

ABSTRACT

In this paper, we study the regenerator location problem (RLP). This problem arises in optical networks where an optical signal can only travel a certain maximum distance (called the optical reach) before its quality deteriorates, needing regenerations by regenerators deployed at network nodes. The RLP is to determine a minimal number of network nodes for regenerator placement, such that for each node pair, there exists a path of which no sub-path without internal regenerators has a length greater than the optical reach. Starting with a set covering formulation involving an exponential number of constraints, reported and studied in Rahman (2012) and Aneja (2012), we study the facial structure of the polytope arising from this formulation, significantly extending known results. Making use of these polyhedral results, we present a new branch-and-cut (B&C) solution approach to solve the RLP to optimality. We present a series of computational experiments to evaluate two versions of the proposed B&C approach. Over 400 benchmark RLP instances, we first compare them with an available exact method for the RLP in the literature. Because of the equivalence among the RLP, the minimum connected dominating set problem (MCDSP), and the maximum leaf spanning tree problem (MLSTP), we further compare our approaches with other available exact algorithms using 47 benchmark MCDSP/MLSTP instances.

1. Introduction

In *all-optical networks*, *optical-bypass* is used to carry the traffic from a source s to a destination t entirely in the optical domain so that no Optical-Electrical-Optical (O/E/O) is required in the intermediate nodes of the path between s and t. Since the optical reach (the maximum distance an optical signal can travel before its quality deteriorates to a level that necessitates regeneration), ranges from 500 to 2000 miles (Simmons, 2006), regeneration of optical signals is essential to establish lightpaths of length greater than the optical reach. In practice, the 3R signal regeneration process is used to re-amplify, reshape and re-time the signal for wide-area backbone networks. Such regenerators are rather expensive equipment (e.g., $160K, see Mertzios, Sau, Shalom, & Zaks (2012)), and much research has been conducted, concerning minimizing their usage while satisfying all or most of the communication requirements posed by the clients. The cost of regenerators in a network is measured in two main ways: (1) the number of regenerators placed in the network, and (2) the number of locations in which regenerators are placed (Hartstein, Shalom, & Zaks, 2013). The second measure is the one that has been used in most of the research on the regenerator location problem (RLP) (Chen, Ljubić, & Raghavan, 2010; Pedrola et al., 2013; Sen, Murthy, & Bandyopadhyay, 2008; Yetginer & Karasan, 2003). The RLP deals with a constraint on the geographical extent of signal transmission in the optical network design. Fig. 1 presents an example of the RLP. In this six-node network, the optical reach is $d_{\max} = 200$ for the signal travel. For this instance, the optimal solution is to place one regenerator at node 4. With this regenerator placement, each node pair can communicate with each other.

In this paper, we focus on the RLP that can be defined as follows.

Definition 1. Given an optical network and the optical reach $d_{\max}$, the RLP is to determine the minimum number of network nodes for regenerator placement, such that for each node pair, there exists a path of which no sub-path without internal regenerators has a length greater than the optical reach $d_{\max}$.

Given an undirected graph $G = (V, E)$, a subset $D \subseteq V$ is a dominating set if every vertex $v \in V \backslash D$ is joined to at least one member of D by an edge in E. A connected dominating set is a dominating set D such that the subgraph $G(D) = (D, E(D))$ is connected,

* Corresponding author.
E-mail addresses: xyli@tongji.edu.cn (X. Li), aneja@uwindsor.ca (Y.P. Aneja).

http://dx.doi.org/10.1016/j.ejor.2016.07.032
0377-2217/© 2016 Elsevier B.V. All rights reserved.

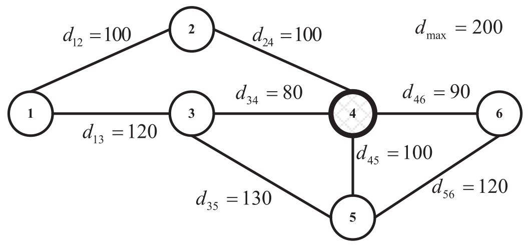

Fig. 1. Example of the RLP.

where $E(D)$ is the set of edges of E with both ends in D(Buchanan, Sung, Boginski, & Butenko, 2014). The minimum connected dominating set problem (MCDSP) consists of finding a connected dominating set of minimum cardinality. The MCDSP is $\mathcal{NP}$-hard since the problem of finding a minimal cardinality dominating set is known to be $\mathcal{NP}$-hard (Garey & Johnson, 1979). It is easy to see, as observed in Gendron, Lucena, Cunha, and Simonetti (2014) and Lucena, Maculan, and Simonetti (2010), that the MCDSP is equivalent to the maximum leaf spanning tree problem (MLSTP), which consists of finding a spanning tree of G with maximum number of leaf nodes. It was shown in Sen et al. (2008) that the RLP is equivalent to the MCDSP, and is hence $\mathcal{NP}$-hard. Hartstein et al. (2013) examined the parameterized complexity of the RLP and presented several fixed parameter tractability results and polynomial algorithms for fixed parameter values, as well as several $\mathcal{NP}$-hardness results.

Since the RLP, MCDSP and MLSTP are equivalent, solution approaches for any one of these problems can be used for the other two problems as well.

Fujie (2004) presented two binary integer programming "noncompact" formulations for the MLSTP — an "edge-vertex" formulation with $|E|+|V|$ binary variables, and a "vertex" formulation with only $|V|$ binary variables, and studied the facial structure of each of these two polytopes obtained by taking the convex hull of feasible solutions of these formulations. However, no separation algorithm or computational experiments were presented in the paper. Many practical applications of the MLSTP/MCDSP (equivalent to the RLP) are discussed in Lucena et al. (2010) and Gendron et al. (2014). Since the problem is $\mathcal{NP}$-hard, researchers have studied exact algorithms (Chen et al., 2010; Fan & Watson, 2012; Fujie, 2003; Gendron et al., 2014; Lucena et al., 2010; Simonetti, Salles da Cunha, & Lucena, 2011), heuristics (Chen et al., 2010; Duarte, Martí, Resende, & Silva, 2014; Lucena et al., 2010; Sen et al., 2008; Yue, Li, Wei, & Lin, 2014), and approximation algorithms (Flammini, Marchetti-Spaccamela, Monaco, Moscardelli, & Zaks, 2011; Guha & Khuller, 1998) for solving these problems.

Given that the investigation focus of this paper is on developing an exact algorithm, we here review works only regarding exact algorithms for the problem. For updated progress of heuristics, please refer to Duarte et al. (2014) and Gendron et al. (2014).

Fujie (2003), based on the "edge-vertex" formulation of the MLSTP discussed in Fujie (2004), presented a branch-and-bound (B&B) algorithm for the problem and reported results of some computational experiments. Lucena et al. (2010) studied two binary integer programming formulations of the MLSTP. The first formulation significantly enhanced the "edge-vertex" formulation of Fujie (2003) by adding some valid inequalities and some facet defining inequalities introduced in Fujie (2004). The second formulation recast the MLSTP as a Steiner arborescence problem on a modified directed graph, and was shown to be computationally superior to the one in Fujie (2003). Chen et al. (2010) studied the RLP, formulated it also as a Steiner arborescence problem on a modified directed graph with a unit degree constraint on the root node, and developed a branch-and-cut (B&C) algorithm. The computational results in Chen et al. (2010) show that their B&C method could optimally solve instances with up to 100 nodes and a few 200- and 300-node instances with small network density. Rahman (2012), and Rahman, Bandyopadhyay, and Aneja (2015) also studied a set covering formulation, which is equivalent to the "vertex" formulation discussed in Fujie (2004). With this formulation, the authors presented a computational study using a preliminary B&C approach (no comparison was made with any other approach).

Simonetti et al. (2011) considered the MCDSP and presented a binary integer programming formulation, using $|V|+|E|$ binary variables, that embedded two structures: one corresponding to the tree polytope for the dominating set of nodes containing the generalized subtour elimination constraints (GSEC), and the other involving the covering constraints corresponding to each node being connected to at least one of the nodes in the dominating set. The authors further strengthened their formulation by lifting both covering constraints and GSEC constraints. Another set of valid cut-inequalities were derived by observing that whenever S and $V\backslash S$ are non-dominating set of nodes, at least one edge across the cut $(S, V\backslash S)$ must be chosen in the solution tree. Fan and Watson (2012) presented compact formulations for the MCDSP and solved these formulations using CPLEX 12.1. Fernau et al. (2011) developed an exact approach, which can solve the MCDSP in $O(1.8966^n)$ time, but did not present computational results. Gendron et al. (2014) investigated further the branch-and-cut algorithm developed by Simonetti et al. (2011), and presented a Benders decomposition algorithm, a branch-and-cut method and a hybrid algorithm combining these two algorithms. The Benders decomposition approach was applied to a formulation involving only $|V|$ binary variables that contained covering constraints as described in Simonetti et al. (2011), and a generic set of constraints imposing connectedness on the selected dominating set. The branch-and-cut algorithm developed in Gendron et al. (2014) was based on strengthening further the formulation in Simonetti et al. (2011) by generalizing the cut-inequalities in Simonetti et al. (2011). Based on 47 benchmark MCDSP/MLSTP instances, Gendron et al. (2014) tested two variants of each of the above three approaches, Benders, B&C and hybrid: a stand-alone version and an iterative probing variant. The computational results showed that the methods in Gendron et al. (2014) are the current best exact approaches for the MCDSP/MLSTP and RLP.

Additionally, Guha and Khuller (1998) provided approximation algorithms for the MCDSP while Flammini et al. (2011) focused on theoretical analysis of regenerator placement, presenting exact algorithms, $\mathcal{NP}$-hardness results, approximation algorithms and hardness of approximation results for various extensions of the RLP. Mertzios et al. (2012) studied a special case of the RLP in which k possible traffic patterns are given and the objective is to place the minimum number of regenerators satisfying each of these patterns. The authors proposed a constant-factor approximation algorithm with ratio $\ln(w\cdot k)$, where w is the maximum allowed number of hops for any lightpath. Additionally, Mertzios, Shalom, Wong, and Zaks (2011) and Shalom, Voloshin, Wong, Yung, and Zaks (2012) studied the regenerator placement in an online setting.

In the computer science and network traffic literature, researchers also studied some problems related to the RLP. Yang and Ramamurthy (2005b) and Pachnicke, Paschenda, and Krummrich (2008) considered the problem of regenerating lightpaths in conjunction with the one of satisfying the greatest possible number of communication requests, given a limited number of wavelengths. Sriram, Griffith, Su, and Golmie (2004) dealt with regenerators with slight different capabilities. Yang and Ramamurthy (2005a) studied the wavelength routing under sparse regeneration in translucent optical networks. Gouveia, Patrício, De Sousa, and Valadas (2003) addressed a multi-protocol label switching (MPLS) over wave division multiplexing (WDM) network design problem in which the path constraint forbids path segments between two components that are longer than a given maximum length. Yetginer and Karasan (2003) also studied the regenerator placement problem in the context of traffic engineering with restoration.

It is well known that integer programming algorithms that employ information about the polyhedral structure of the problem's associated polytope, are typically more effective for solving large instances than those that ignore this structure (Mitchell, 2009). Adding strong cutting planes (hopefully facet-defining inequalities) makes it possible to reduce the size of the branch-and-bound tree significantly, thus enabling the procedure to find optimal solutions to much larger instances of the problem.

In this paper, we study the associated polyhedral structure, starting with a set covering formulation, and develop new efficient branch-and-cut approaches. We study the polyhedral structure of the polytope of the convex hull of all feasible solutions, including: (1) derive necessary and sufficient conditions under which our constraints are facet defining, (2) provide a method to lift non-facet-defining inequalities to get stronger "lifted inequalities", (3) present necessary and sufficient conditions under which these lifted inequalities are facet defining, and (4) give an approach to lift these non-facet-defining lifted inequalities. Our first main result provides simple necessary and sufficient conditions for a set covering inequality to be a facet-defining inequality. Although this result is equivalent to the result in Fujie (2004), our conditions lead to an efficient method to test if a valid inequality obtained through the separation routine is facet-defining, and lifting it to one or a set of stronger inequalities if it is not facet-defining. With the proposed polyhedral results, we propose two versions of our B&C approach to solve the RLP to optimality: one using derived valid inequalities without lifting, and another using lifted inequalities. A series of experiments are finally carried out to evaluate the performance of the two versions of our B&C approach by comparing them with existing algorithms in the literature. Using 400 benchmark RLP instances, we first compare our approach with the one used in Chen et al. (2010). Because the RLP is equivalent to the MCDSP and the MLSTP, we then compare our approach with nine other exact algorithms using 47 benchmark MCDSP/MLSTP instances. The computational results demonstrate that our proposed approach outperforms other existing algorithms, especially for larger instances. The proposed approach extends our ability to solve large RLP instances with up to 500 nodes. Additionally, our computational results also demonstrate that our B&C approach does benefit from lifted cuts, resulting in a significant saving of running time.

We organize the remainder of this paper as follows. In Section 2, we study an integer programming formulation, which is equivalent to the one given in Fujie (2004). In Section 3, we study the polyhedral structure of the RLP. In Section 4, we detail methods for identifying violated cuts in the B&C approach. In Section 5, we highlight the B&C solution approach. In Section 6, we report computational results. Finally, we summarize our work in Section 7.

2. Formulation of the RLP

We first define the RLP on an undirected graph $G = (V, F)$, where $V = \{1, 2, \dots, n\}$ denotes the set of n vertices and F is the set of all available edges. With each edge $\{i, j\} \in F$, we associate a distance $d_{i,j}$. A maximum distance limit of d_{max} specifies how far an optical signal can travel before its quality deteriorates, needing regeneration. The RLP can be formally restated as follows:

Definition 2. Determine a minimum cardinality subset L of regenerator equipped nodes such that for every pair of nodes in G there exists a path with the property that there is no sub-path with

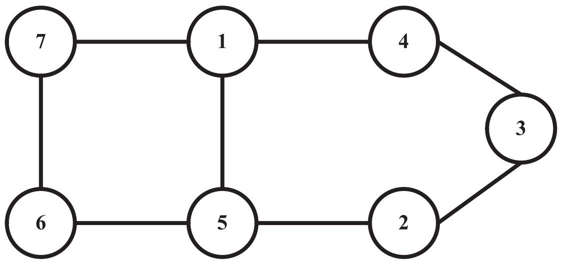

Fig. 2. Example of DSNs.

distance more than $d_{\max}$ without internal regenerators (i.e., we do not include the end-nodes of the sub-path).

As in Sen et al. (2008), we also define the formulation on a transformed graph $M = (V, E)$ generated from the original graph G. Hereafter, we refer to this transformed graph as the reachability graph. Given graph G and a maximum distance of $d_{\max}$, the reachability graph M is created as follows. We first apply the "all pairs shortest path algorithm" on graph G, and we then keep an edge $\{i, j\} \in G$ in E only if the shortest distance between nodes i and j in G is less than or equal to the distance limit $d_{\max}$. Obviously, if M is a complete graph, no regenerators are needed for the RLP instance. Otherwise, every node pair that is not connected by an edge in M requires at least one regenerator to communicate with each other. Therefore, we can equivalently restate the RLP on graph M as follows:

Definition 3. Find a minimum cardinality subset L of regenerator equipped nodes such that for every not-directly-connected (NDC) node pair $\langle s, t\rangle$ in M, there exists a path between s and t such that all its internal nodes are regenerator equipped.

Since, the RLP is equivalent to the MCDSP, we give below another equivalent definition of the RLP, as we use this sometimes for our analysis.

Definition 4. Find a minimum cardinality subset of L of regenerator equipped nodes such that subgraph induced by these nodes is connected, and every other node is connected to at least one of these regenerator equipped nodes.

Define D to be a "Disconnecting Set of Nodes" (DSN) if removal of these nodes and all edges adjacent to these nodes from the reachability graph M disconnects at least one pair of nodes in M. We refer to D as a minimal DSN if no proper subset of D is a DSN. Fig. 2 gives an example of DSNs. In this seven-node reachability graph, {1, 4, 5} is a DSN, not a minimal DSN because its subset {4, 5} is a DSN. Furthermore, subsets {4, 5} and {1, 5} are both minimal DSNs.

In the following, we introduce the formulation of the RLP. Let Ψ be the set of all minimal DSNs defined on graph M. With each node $i \in V$, we associate a binary variable x_i that equals 1 if node i is equipped with a regenerator, and 0 otherwise. Lemma 1, from Rahman (2012), gives necessary and sufficient conditions for x to correspond to a feasible solution for the RLP .

Lemma 1. *Any binary solution $x = (x_1, x_2, \dots, x_j, \dots, x_n)$ defines a feasible solution to a RLP instance if and only if it satisfies constraints*

$$\sum_{i\in D} x_i \geq 1 \quad \forall D \in \Psi. \tag{1}$$

Proof. The "only if" part is obvious. Consider the "if" part. Suppose we have a solution $\bar{x}$ that satisfies constraints (1), but is not a feasible solution. That is, there exists at least one node pair i and j that is not connected in M with regenerator placement defined by $\bar{x}$. We can determine the set W_i of nodes that can communicate with node i. Obviously, we have $j \notin W_i$. We further define D_i as the set of nodes in W_i with $\bar{x}_i = 0$. If we want to ensure that node i can communicate with node j, there must be at least one node $i \in D_i$ with regenerator equipped. Thus D_i should be a DSN and have been covered by constraints in (1). This contradicts the given assumption. □

Based on this observation, the following set covering formulation was given in Rahman (2012) and also published in Rahman et al. (2015).

$$\min \quad \sum_{i \in V} x_i$$

$$\text{subject to } \sum_{i \in D} x_i \geq 1 \quad \forall D \in \Psi, \tag{2}$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \tag{3}$$

The above formulation is equivalent to the “vertex” formulation in Fujie (2004) for the MLSTP: there is a linear transformation $y_j = 1 - x_j$, that converts the set-covering formulation to the “vertex” formulation in Fujie (2004), and vice versa. Note that the node-based integer linear programming model has been used for other applications including Chen, Ljubić, and Raghavan (2015), Fügenschuh and Fügenschuh (2008), Fischetti et al. (2014), and Álvarez-Miranda, Ljubić, and Mutzel (2013).

3. Polyhedral structure

Let polytope $\mathcal{P}$ be the convex hull of all feasible solutions to the RLP. We next study the polyhedral structure of the convex hull of the RLP that essentially establishes a theoretical explanation for the success of our proposed approach in Section 5. A preliminary study of the facial structure of this polytope was presented in (Aneja, 2012). Without loss of generality, we can assume that graph M is bi-connected. If graph M is not bi-connected, we must place regenerators at all of its cut-vertices.

Lemma 2. *$\mathcal{P}$ is full-dimensional.*

Proof. The n-vectors $\{\mathbf{1} - e_i : i = 1, \ldots, n\}$, and the 1-vector (vector of all 1s) are all affinely independent feasible solutions to the RLP. □

Lemma 3. *For any node $i \in V$, $x_i \leq 1$ is a facet-defining inequality (FDI).*

Lemma 4. *For any node $i \in V$, $x_i \geq 0$ is an FDI if and only if $M \backslash \{i\}$ is bi-connected.*

We refer to such inequalities ($x_i \geq 0$, $x_i \leq 1$) as trivial inequalities. It is well known that any non-trivial *FDI* of a set covering formulation is of the form $a^T x \geq b$, or

$$\sum_{i \in V} a_i x_i \geq b, \tag{4}$$

where $a_i \geq 0$ $\forall i \in V$, and $b > 0$ (Balas & Ng, 1989).

Theorem 1 was established in Agarwal and Aneja (2012). We will make extensive use of this theorem for our following facet proofs.

Theorem 1. *Consider a non-trivial valid inequality (4), where k is the number of non-zero a_i-values. Assume that $a_i > 0$ $(i = 1, \ldots, k)$, and $a_i = 0$ $(i = k+1, \ldots, n)$. It is an FDI if and only if the following two conditions are satisfied: (a) there exist k 0–1 feasible n-vectors that satisfy (4) as an equality, and k sub-vectors (each sub-vector consisting of the top k elements of these k vectors) are linearly independent, and (b) for each j $(j = k+1, \ldots, n)$, there exists a feasible solution x with $x_j = 0$, and satisfies (4) as an equality.*

Because a non-minimal DSN cannot correspond to an FDI, we assume from here on that any DSN D under consideration is minimal. Consider one such D. For each node $i \in D$, we first define a graph M_i that is generated by removing, from graph M, all nodes $j \in D \backslash \{i\}$ and all of the edges incident to nodes $j \in D \backslash \{i\}$. Define S_i to be the set of all cut-vertices of M_i. The set S_i can be identified in $O(|E|)$ (Reingold, Nievergelt, & Deo, 1977).

Let $C = \bigcap_{i \in D} S_i$. Note that $C \bigcap D = \emptyset$. The following theorem presents a necessary and sufficient condition for D to correspond to an FDI.

Theorem 2. *$\Sigma_{i \in D} x_i \geq 1$ is an FDI if and only if $C = \emptyset$.*

Proof. Consider first the “if” part of the theorem. Thus assume that $C = \emptyset$. Let $|D| = k$, and assume $D = \{1, 2, \ldots, k\}$. Let us first prove that condition (a) of Theorem 1 is satisfied. Take a node $i \in D$. Since D is minimal, by assumption, removal of all the nodes in $D \backslash \{i\}$ from M does not disconnect M. That is, graph M_i is connected, and putting a regenerator at each of the nodes in M_i provides a feasible solution to the RLP. Thus, for each $i \in D$, create an n-vector x^i such that

$$x_r^i = \begin{cases} 0, & \forall r \in D \backslash \{i\}, \\ 1, & \text{otherwise.} \end{cases}$$

Each of these k vectors corresponds to a feasible solution and satisfies $\sum_{i \in D} x_i = 1$. Consider the $n \times k$ matrix of these k feasible solutions. Since the top k rows of this matrix correspond to the k elements of D, the $k \times k$ submatrix of these top k rows is an identity matrix, and hence these k sub-vectors are linearly independent. Thus, condition (a) of Theorem 1 is satisfied. Consider, now, condition (b) of Theorem 1. Take a node $j \notin D$. Suppose we do not allow a regenerator to be placed at this node. Since $C = \emptyset$, there is some $i \in D$ such that $j \notin S_i$. Create an n-vector x^j:

$$x_r^j = \begin{cases} 0, & \forall r \in D \cup \{j\} \backslash \{i\}, \\ 1, & \text{otherwise.} \end{cases}$$

It is easy to verify that these $n - k$ vectors $\{x^j : j \in V \backslash D\}$ are feasible, linearly independent, and satisfy $\sum_{i \in D} x_i = 1$. Thus condition (b) of Theorem 1 is also satisfied.

Consider now the “only if” part of the theorem. Suppose, $j \in C$. That is, there is a node $j \in V \backslash D$ such that node j is a cut-vertex for graph M_i $\forall i \in D$. Thus, placing a regenerator only at one of the nodes in D, requires a regenerator at node j to obtain a feasible solution. That is, $x_j = 1$ for every feasible solution that satisfies $\sum_{i \in D} x_i = 1$. Hence condition (b) of Theorem 1 is not satisfied. □

As we pointed out earlier, since the RLP and MLSTP are equivalent, Theorem 2 can be shown, with some effort, to be equivalent to Theorem 3.9 in Fujie (2004).

Consider the example in Fig. 2 to illustrate this theorem. In this example, consider the minimal DSN: $D = \{1, 2\}$. Now $S_1 = \{1, 4\}$, and $S_2 = \{2, 3, 5, 6\}$. Hence $C = S_1 \bigcap S_2 = \emptyset$. Seven feasible, linearly independent vectors that satisfy $x_1 + x_2 = 1$, based on the construction in the proof, are given below:

Node	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	1	0	1	0	1	1	1
2	0	1	0	1	0	0	0
3	1	1	0	1	1	1	1
4	1	1	1	0	1	1	1
5	1	1	1	1	0	1	1
6	1	1	1	1	1	0	1
7	1	1	1	1	1	1	0

It follows from the above theorem that if $\bigcap_{i \in D} S_i \neq \emptyset$, then $\Sigma_{i \in D} x_i \geq 1$ is not an FDI. In this case, we show that this inequality can be lifted, resulting in one or a set of stronger inequalities.

Lemma 5. *Suppose $C \neq \emptyset$, for a given D. Then $\sum_{i\in D} x_i + x_j \geq 2$ is a valid inequality for $\mathcal{P}$, for each node $j \in C$.*

Proof. Note that if $x_j = 1$, then the above inequality reduces to the valid inequality: $\Sigma_{i \in D} x_i \geq 1$. If $x_j = 0$, then, because node $j \in C$, we cannot find a feasible solution if we equip only one node of D with a regenerator. Hence $\Sigma_{i \in D} x_i \geq 2$ is also a valid inequality in this case. □

We now provide necessary and sufficient conditions for such a lifted inequality to be an FDI. Consider, for any node $j \in C$, the corresponding lifted valid inequality: $\sum_{i\in D} x_i + x_j \geq 2$. Let $\widehat{M}_j$ be the graph obtained after removing node j and all its incident edges from graph M. Let $\widehat{S}_j$ be the set of all the cut-vertices of $\widehat{M}_j$.

Theorem 3. *Consider a node $j \in C$. Then $\sum_{i\in D} x_i + x_j \geq 2$ is an FDI if and only if $|D \bigcap \widehat{S}_j| \leq 2$ and $|\{D \bigcup \{j\}\} \bigcap \widehat{S}_k| \leq 2$ for all $k \in C \backslash \{j\}$.*

Proof. We again make use of the two conditions of Theorem 1 for the proof of this theorem. Consider the "if" part, first. To prove condition (a) of Theorem 1, we produce $|D| + 1$ affinely independent feasible solution vectors that satisfy $\sum_{i\in D} x_i + x_j = 2$. For each $i \in D$, define vector x^i as in the proof of Theorem 2. That is,

$$x_r^i = \begin{cases} 0, & \forall r \in D\backslash\{i\}, \\ 1, & \text{otherwise.} \end{cases}$$

Now define vector x^j as follows. First set $x_j^j = 0$ and $x_r^j = 1$ for all $r \in V \backslash \{D \cup \{j\}\}$. Now set $x_r^j = 1$ for all $r \in D \bigcap \widehat{S}_j$. If $|D \bigcap \widehat{S}_j| < 2$, then pick any $2 - |D \bigcap \widehat{S}_j|$ nodes from $D \backslash \widehat{S}_j$ and set their x-value to 1. Set x-value for all other nodes in D to 0. Note that x^j is a feasible solution to the RLP since all cut-vertices of D in $\widehat{M}_j$, and all nodes in $V \backslash \{D \cup \{j\}\}$ are equipped with regenerators. Further, each of these $|D| + 1$ vectors $\{x^i\colon i \in D, x^j\}$ satisfies $\sum_{r\in D} x_r + x_j = 2$. Finally, because $x_j^i = 1$ for all $i \in D$, and $x_j^j = 0$, these $|D| + 1$ vectors are affinely independent. Hence condition (a) of Theorem 1 is satisfied.

To satisfy condition (b) of Theorem 1, first consider any node $k \notin D \cup C$. Since $k \notin C$, there is a node $i \in D$ such that $k \notin S_i$. Create an n-vector x^k:

$$x_r^k = \begin{cases} 0, & \forall r \in D \bigcup \{k\} \backslash \{i\}, \\ 1, & \text{otherwise.} \end{cases}$$

It is easy to see that x^k is a feasible vector with $\sum_{i\in D} x_i^k + x_j^k = 2$. Consider, finally, any node $k \in C \backslash \{j\}$. We construct vector x^k as follows. Its construction is quite similar to the construction of vector x^j. First, set $x_k^k = 0$. Then set $x_i^k = 1$ for all $i \in V \backslash \{D \cup \{j, k\}\}$, and all $i \in \{D \cup \{j\}\} \bigcap \widehat{S}_k$. If $|\{D \cup \{j\}\} \bigcap \widehat{S}_k| < 2$, then select any $2 - |\{D \cup \{j\}\} \bigcap \widehat{S}_k|$ nodes of $D \bigcup \{j\} \backslash \widehat{S}_k$ and set their x-value to 1. All other nodes of D are assigned an x-value of 0. Note that with this construction, exactly 2 nodes of $D \cup \{j\}$ are equipped with a regenerator, i.e. assigned an x-value of 1. The vector corresponds to a feasible solution of the RLP. It is easy to verify that such constructed $n - (|D| + 1)$ vectors (one for each node in $V \backslash \{D \bigcup \{j\}\}$) are feasible solutions, and each one satisfies $\sum_{i\in D} x_i^k + x_j^k = 2$. Thus condition (b) of Theorem 1 is also satisfied.

To prove the "only if" part, note that if $|D \bigcap \widehat{S}_j| > 2$, then setting $x_j = 0$, i.e. removing node j and incident nodes from M, results in at least three nodes of the DSN being cut-vertices, and all of these must be equipped with a regenerator to get a feasible solution. Further, if $|\{D \cup \{j\}\} \bigcap \widehat{S}_k| > 2$ for any $k \in V \backslash \{D \bigcup \{j\}\}$, then $x_k = 1$ for any feasible solution that satisfies: $\sum_{i\in D} x_i + x_j = 2$. Hence, condition (b) of Theorem 1 will be violated. □

Consider again the example in Fig. 2 to illustrate Lemma 5 and Theorem 3. In this example, consider the minimal DSN: $D = \{1, 5\}$. Now $S_1 = \{1, 3, 4, 7\}$, and $S_5 = \{2, 3, 5, 6\}$. Hence $S_1 \bigcap S_2 = \{3\}$ and $x_1 + x_5 \geq 1$ is not an FDI. To show that $x_1 + x_3 + x_5 \geq 2$ is valid, we consider $D_1 = \{1, 5\}$, $D_2 = \{1, 3\}$, and $D_3 = \{3, 5\}$. All these three sets are DSNs. Hence we have $x_1 + x_5 \geq 1$, $x_1 + x_3 \geq 1$, and $x_3 + x_5 \geq 1$. Adding them and dividing by 2, we have $x_1 + x_3 + x_5 \geq \frac{3}{2}$ that implies $x_1 + x_3 + x_5 \geq 2$. This lifted inequality is also an FDI. To show this, seven feasible linearly independent vectors that satisfy $x_1 + x_3 + x_5 = 2$, based on the construction in the proof, are given below:

Node	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	1	1	1	1	0	1	1
2	1	0	1	1	1	1	1
3	1	1	0	0	1	0	0
4	1	1	1	0	1	1	1
5	0	0	1	1	1	1	1
6	1	1	1	1	1	0	1
7	1	1	1	1	1	1	0

It follows from Theorem 3 that if $j \in C$, and $|D \bigcap \widehat{S}_j| > 2$, or $|\{D \bigcup \{j\}\} \bigcap \widehat{S}_k| > 2$, for any $k \in C \backslash \{j\}$, then $\sum_{i\in D} x_i + x_j \geq 2$ is not an FDI. In that case, the inequality can then be further lifted. We state the following results without proof.

Lemma 6. *If $j \in C = \bigcap_{i\in D} S_i$, and $|D \bigcap \widehat{S}_j| = p > 2$, then $\sum_{i\in D} x_i + (p-1)x_j \geq p$ is a valid inequality.*

Lemma 7. *If $j \in C = \bigcap_{i\in D} S_i$, and $|\{D \bigcup \{j\}\} \bigcap \widehat{S}_k| = p > 2$ for a $k \in V \backslash \{D, j\}$, then $\sum_{i\in D} x_i + x_j + (p-1)x_k \geq p$ is a valid inequality.*

4. Separation method for violated cuts

The set covering formulation involves an exponential number of constraints. Thus it is impractical to explicitly enumerate all these constraints in the starting formulation. Our proposed B&C approach starts to enumerate a few simple DSNs in Ψ to initialize the formulation, and adds valid cuts violated by the LP solution $\bar{x}$ at each node in the branch-and-bound tree. The separation problem is to find a DSN D, if one exists, such that $\sum_{i\in D} \bar{x}_i < 1$.

As we will show below, a valid DSN D corresponds to a cut in a capacitated reachability digraph. Therefore, solving the separation problem is equivalent to finding a minimum cut over a properly modified reachability graph. Given a fractional solution $\bar{x}$ at each node of the enumeration tree, we here discuss a min-cut based method for solving the separation problem exactly. Note that the separation of DSN inequalities using maximum flows follows from the Max-flow Min-Cut Theorem (Ford & Fulkerson, 1974), reported in Rahman (2012); Rahman et al. (2015), which is also used in Chen et al. (2015), Fügenschuh and Fügenschuh (2008), Fischetti et al. (2014), and Álvarez-Miranda et al. (2013). If the identified valid cut is not an FDI, we further lift it to a stronger inequality, or a set of stronger inequalities. The detailed implementation of this separation method is presented in Section 5.3.

Based on graph $M = (V, E)$, we first define a capacitated reachability digraph $M' = (V', A')$. For each node $i \in V$, we add two nodes i and i' to V', i.e., $V' = \{\{i, i'\} : i \in V\}$. The arc set A' is defined by $A' = \{\{(i, i'), (i', j), (j, j'), (j', i)\} : \{i, j\} \in E\}$. Fig. 3 gives an example of how to generate a capacitated reachability digraph. Fig. 3(a) gives graph M with four nodes and five edges. The returned capacitated digraph M' in Fig. 3(b) has eight nodes and 14 arcs. With such a transformation, there is a correspondence between a path connecting each NDC node pair in (a) and a directed path in (b). For example, path $1-2-4$ connecting nodes 1 and 4 corresponds to path $1' \to 2 \to 2' \to 4$ in (b) of Fig. 3. Let Γ be the set of all the NDC node pairs in E.

In a capacitated reachability graph, a cut $[\Omega, V' \backslash \Omega]$ separating i' and j is defined as a partition of the node set V' into two nonempty parts Ω and $V' \backslash \Omega$ with $i' \in \Omega$ and $j \in V' \backslash \Omega$. Any finite capacity

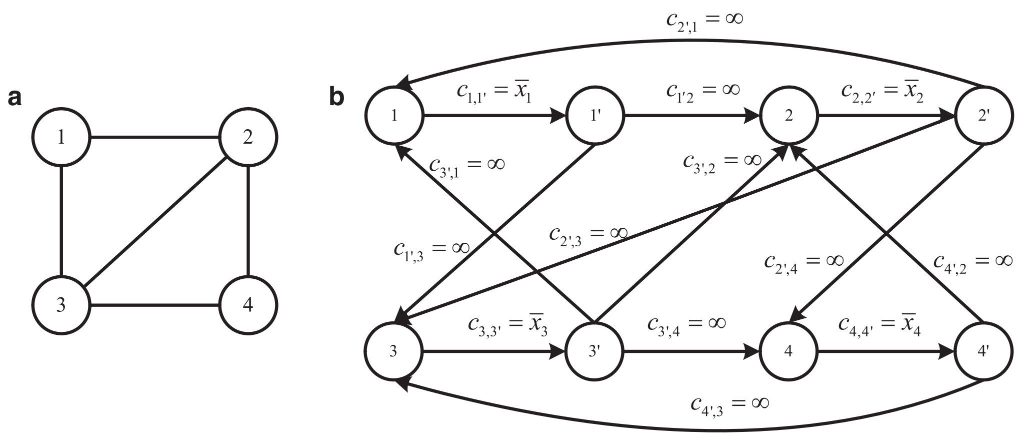

Fig. 3. Transforming a reachability graph to a capacitated reachability digraph.

cut of $[\Omega, V'\backslash\Omega]$ consists only of arcs of the kind (i, i'), and hence define a valid DSN $D = \{i : i \in \Omega, i' \in V'\backslash\Omega\}$.

For example, in Fig. 3(b), cut $[\Omega, V'\backslash\Omega]$ with $\Omega = \{1, 1', 2, 3\}$ corresponds to a DSN $D = \{2, 3\}$. Following this equivalence, the separation problem of finding a valid DSN D in the reachability graph M reduces to finding a proper cut in the capacitated reachability graph M'. Hence, every DSN D corresponds to a cut $[\Omega, V'\backslash\Omega]$ in the reachability graph.

Additionally, the arc capacity function on M' is defined as follows: capacity $c_{i,i'} = \bar{x}_i$ for arc (i, i') and $c_{i,j} = \infty$ for other arcs $(i, j) \in A'$. Using the equivalence between a cut and a DSN, we now describe how to solve the separation problem. Viewing each node i' as a source node and node j as a sink node, we solve a maximum flow problem in the capacitated reachability graph M' and we then get the corresponding minimum cut $[\Omega, V'\backslash\Omega]$ separating source node i' and sink node j. The capacity of this cut is computed as $\sum_{i\in\Omega:i'\in V'\backslash\Omega}\bar{x}_i$. If the cut capacity is less than 1, then we have a violated valid inequality (valid DSN D) as

$$\sum_{i\in\Omega:i'\in V'\backslash\Omega} x_i \geq 1. \tag{5}$$

Hereafter, we refer to any cut $[\Omega, V'\backslash\Omega]$ with $s' \in \Omega$ and $t \in V'\backslash\Omega$ as an $s-t$ cut, and refer to the problem of determining a minimum capacity $s-t$ cut as the minimum $s-t$ cut problem. To solve the separation problem, we need to solve one minimum $s'-t$ cut problem for each NDC node pair $\langle s, t\rangle \in \Gamma$. In our proposed B&C approach, we use the Pseudoflow method to solve each minimum $s-t$ cut problem (Chandran & Hochbaum, 2009; Hochbaum, 2008), and then add the violated inequality to the LP subproblem, if one exists.

As stated above, we need to solve $|\Gamma|$ minimum $s-t$ cut problems to completely solve the separation problem. In certain cases, $|\Gamma|$ could be quite large. Using Lemma 8, we can significantly reduce the number of minimum $s-t$ cut problems solved for the separation problem. Without loss of generality, we assume that in the reachability graph M, node 1 is a node with minimum degree. Let Λ be the set of all the nodes adjacent to node 1, including node 1, in graph M, i.e., $\Lambda = \{j : \{1, j\} \in E\} \cup \{1\}$. We refer to the minimum weighted DSN problem as determining a DSN D such that $w(D) = \sum_{i\in D}\bar{x}_i$ is minimized. The corresponding DSN D is referred to as optimal DSN. We want to show that there is an optimal DSN that does not include all the nodes in Λ.

Lemma 8. *The minimum weighted DSN problem can be solved by solving a series of minimum cut problems, considering only NDC node pairs $\langle s, t\rangle$ with either node s or node t in Λ.*

Proof. First note that $\bar{x}_j \geq 0\ \forall j \in V$. Let D^* be an optimal DSN. If $\Lambda \subset D^*$, then there is nothing to prove. So assume that $\Lambda \subset D^*$. Then $w(\Lambda\backslash\{1\}) \leq w(D^*)$. Further $\Lambda\backslash\{1\}$ is a valid DSN as it separates node 1 from the rest of the graph. That is, $w(\Lambda\backslash\{1\}) \geq w(D^*)$. Hence $w(\Lambda\backslash\{1\}) = w(D^*)$. □

As a result of Lemma 8, we can significantly reduce the size of the separation problem to a moderate size and thus enhance the computational efficiency of our proposed approach.

In the next section, we will discuss the detailed implementation of our separation method.

5. Branch-and-cut approach

With the separation method given in Section 4, we now detail the proposed branch-and-cut approach. The B&C is a generalization of the B&B where the LP relaxation problem is solved to obtain a lower bound at each node in the B&B search tree, and possible cutting planes are identified to tighten the LP bounds (Gendron, Scutellà, Garroppo, Nencioni, & Tavanti, 2016; Wolsey, 1998). If a node has a fractional solution and cannot be pruned, we try to find valid inequalities violated by the current LP solution. If no cut is found, the branching is performed to create new children nodes in the B&B tree. If the LP solution is integer, but does not produce a feasible solution, a violated valid inequality is also identified and added to the LP subproblem.

We develop two versions of the B&C approach:

(1) B&C1: the B&C approach with the min-cut based separation method for identifying violated cuts.
(2) B&C2: the B&C approach in which we first check whether a valid cut found by the separation method corresponds to an FDI and if not, then lift it to one or a set of stronger inequalities.

We now detail the main components of the two versions of our proposed B&C approach.

5.1. Starting formulation

To start with, we enumerate a few simple DSNs in Ψ to initialize the starting formulation as follows. For each node $i \in N$, we first produce a DSN $D = \{j \in N : \{i, j\} \in E\}$ if $|D| \neq n-1$. Using these simple DSNs, we then get the corresponding constraints (2) in the initial formulation.

5.2. Starting incumbent

A starting incumbent might eliminate some parts of the solution space and thus accelerates the solution process of an exact algorithm. In the RLP, a feasible solution corresponds to a spanning tree with all internal nodes equipped with regenerators. In

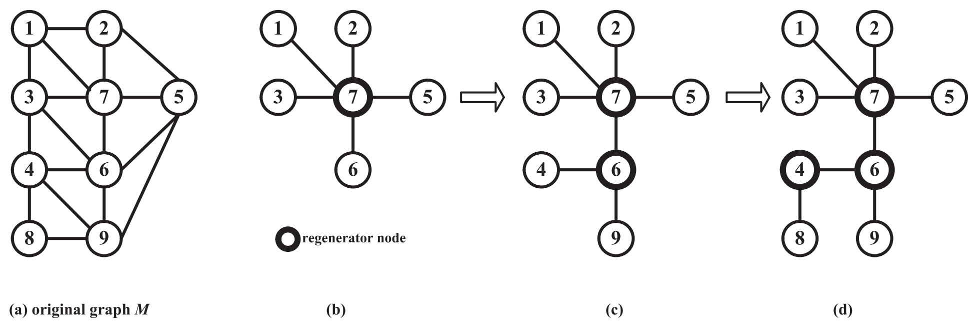

Fig. 4. Producing a starting incumbent from reachability graph M with threshold $\xi = 0.25$.

our B&C approach, we used Procedure 1 to generate a starting incumbent that heuristically selects edges and appends them to a partial tree T until all of the nodes are included.

Before giving a detailed description of Procedure 1, we first introduce some notation. Let T be a partial tree for generating feasible solution. Let d_i be the degree of node $i \in V$ in graph M. For each leaf node i in T, let p_i be the number of NDC node pairs that can be connected when we deploy a regenerator at node i. We refer to an edge $\{i, j\}$ as a desirable edge if appending it to the partial tree T does not produce a cycle and one of its endpoints is in T. For each node $i \in T$, let w_i be the number of desirable edges $\{i, j\}$ that can be appended to the partial tree T. Let $LF(T)$ be the set of leaf nodes in T. Let #V be the number of NDC node pairs that is able to communicate with each other in terms of regenerator placement in T.

Procedure 1 is a threshold based greedy heuristic where parameter ξ is a threshold, and works as follows. First node i^* with maximum p_{i^*} is chosen as regenerator node and edges incident to node i^* are used to initialize tree T. Then, at each subsequent iteration, if no more than $(|\Gamma|\xi)$ NDC node pairs can be connected in terms of the current regenerator placement, leaf node $i \in T$ with maximum p_i is always selected as regenerator node. Otherwise, Procedure 1 chooses as regenerator node $i \in T$ with maximum w_i. Given this spanning tree T, we produce a feasible solution x by setting $x_i = 1$ for all internal nodes i.

Fig. 4 presents an example of producing a feasible solution for reachability graph M. Suppose threshold $\xi = 0.25$. At the beginning, we have $p_8 = 0,\ p_1 = p_2 = 1,\ p_3 = p_4 = p_5 = p_9 = 3$, and $p_6 = p_7 = 5$. We first choose to locate a regenerator at node 7 and initialize the tree T (see Fig. 4(b)). The current $\#V/|\Gamma| = 0.263$ is greater than the threshold. We then choose node 6 as the new regenerator, which has the maximum w_i (e.g., $w_1 = w_2 = 0,\ w_3 = w_5 = 1$ and $w_6 = 2$) and extend the spanning tree (see Fig. 4(c)). Finally we place a regenerator at node 4 and produce a feasible solution with three regenerators.

Procedure 1. Producing starting incumbent

```
input instance data.
set $x = (0, 0, \ldots, 0)$.
compute $p_i$ and set $w_i = d_i, \forall i \in V$.
select one node $i^*$ with maximum $p_{i^*}$.
set $L = \{i^*\}$, $L' = \{j : \{i^*, j\} \in E, j \in V\}$, and $T \leftarrow T \bigcup \{\{i^*, j\} : j \in L'\}$.
update $LF(T)$, $p_i, w_i, \forall i \in LF(T)$, and #V.
while ($L' \neq V$) do
    if (#V $\le \xi|\Gamma|$) do
        choose node $i^* \in LF(T)$ with maximum $p_{i^*}$.
    else
        select node $i^* \in LF(T)$ with maximum $w_{i^*}$.
    end if
    set $L \leftarrow L \cup \{i^*\}$ and $L_1 = \{j : \{i^*, j\} \in E \backslash T, j \in V\}$.
    set $L' \leftarrow L' \cup L_1$ and $T \leftarrow T \cup \{\{i^*, j\}: j \in L_1\}$.
    update $LF(T)$, $p_i, w_i, \forall i \in LF(T)$, and #V.
end while
set $x_i = 1$ for $i \in L$.
return solution $x$.
```

The threshold ξ affects the performance of Procedure 1. Using 200 instances, we will show in Section 6 that, on average, Procedure 1 with greater threshold produces better starting solution. But it does not necessarily produce the best solution in each instance. Therefore, in our B&C approach, we run Procedure 1 with different thresholds and choose the best resulting solution for each instance. Note that Procedure 1 can also be implemented as a simple heuristic for solving the RLP. Our preliminary results show Procedure 1 worked better than the three heuristics that were presented in a recent conference (Yue et al., 2014). In Section 6, we observe that as a simple heuristic, Procedure 1 performs very well and works quite fast.

5.3. Identification of valid inequalities

Section 4 describes how a violated inequality is identified based on a fractional LP solution $\bar{x}$. At each node of the B&B tree, the solution $\bar{x}$ to the LP subproblem may be integral or fractional. We now present the detailed implementation of the separation method for these two cases.

5.3.1. Case 1: integral solution $\bar{x}$

We first deal with the case when $\bar{x}$ is integral. This integral solution defines a regenerator placement: each node i with $\bar{x}_i = 1$ is a regenerator node. We need to verify if this regenerator placement is feasible. That is, we check whether each NDC node pair can be connected with the regenerator placement defined by solution $\bar{x}$. Using logic similar to the one in Section 5.2, we verify the feasibility of $\bar{x}$ as follows. First, for $\bar{x}$ to be feasible, it is necessary that the subgraph induced by regenerator equipped nodes is connected. Second, every other node (each node i for which $\bar{x}_i = 0$), must be connected to at least one of these regenerator equipped nodes. If both these conditions are satisfied, $\bar{x}$ is a feasible solution. Otherwise, the solution is infeasible and a violated valid inequality is added as follows.

Let $R = \{i : \bar{x}_i = 1\}$ and $\bar{R} = V \backslash R$. If the subgraph induced by regenerator nodes is not connected, this immediately implies a violated inequality:

$$\sum_{i \in \bar{R} : \exists j \in R, \{i,j\} \in E} x_i \ge 1. \tag{6}$$

If some node i^* is not connected to at least one regenerator node, this implies the following violated valid inequality:

$$\sum_{i \in \bar{R} : \{i, i^*\} \in E, \exists j \in R, \{i,j\} \in E} x_i \ge 1. \tag{7}$$

In the two versions of our B&C approach, we only add one violated cut if there are more than two nodes that are not connected to at least one regenerator node.

5.3.2. Case 2: fractional solution $\bar{x}$

In the context of fractional $\bar{x}$, we implement the min-cut based method proposed in Section 4 to identify valid inequalities.

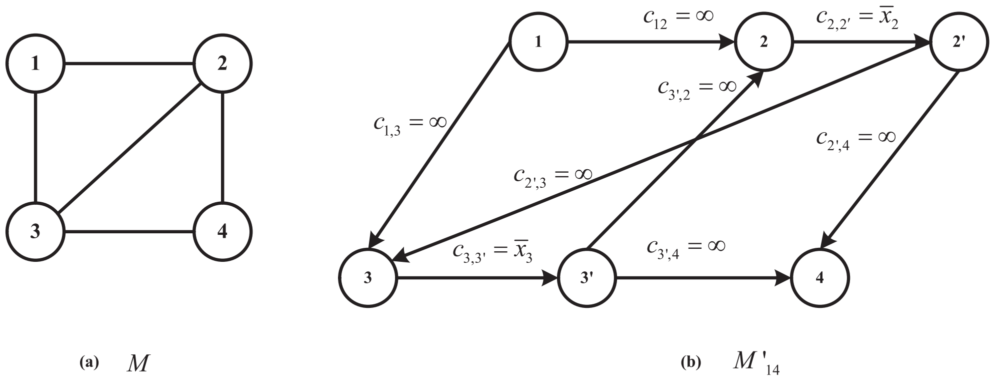

Fig. 5. Generating capacitated reachability graph for a cut .

Using results in Lemma 8, we implement our separation method for fractional solution. Suppose we want to find a minimum cut separating nodes s and t in M. We can assume that all arcs go out of node s and all arcs go into node t in M'. That is, we need not create node s' and t' while producing graph M' from M. Associated with each cut $s-t$, we define the corresponding capacitated reachability graph as $M'_{st}=(\widehat{V},\widehat{A})$. The node set $\widehat{V}$ and arc set $\widehat{A}$ are respectively defined as $\widehat{V}=\{\{i,i'\}:i\in V\backslash\{s,t\}\}\bigcup\{s,t\}$ and $\widehat{A}=\widehat{A}_1\bigcup\widehat{A}_2\bigcup\widehat{A}_3$ with

$$\widehat{A}_1=\{\{(i,i'),(i',j),(j,j'),(j',i)\}:\{i,j\}\in E, i,j\in V\backslash\{s,t\}\},$$
$$\widehat{A}_2=\{\{(i,i'),(s,i)\}:\{i,s\}\in E\}, \text{and}$$
$$\widehat{A}_3=\{\{(i,i'),(i',t)\}:\{i,t\}\in E\}.$$

Consider Fig. 5(a) as an example to show how to create M'_{st}. Suppose we want to find a cut separating nodes 1 and 4. The capacitated reachability graph M'_{14} is given in Fig. 5(b).

After creating the capacitated reachability graph M'_{st}, we then implement the Pseudoflow method to solve the minimum $s-t$ cut problem (Chandran & Hochbaum, 2009; Hochbaum, 2008). If the capacity of the returned minimum cut $[\Omega,\widehat{V}\backslash\Omega]$ is less than one, we then have a valid DSN $D=\{i\in\Omega: i'\in\widehat{V}\backslash\Omega\}$, which defines a violated valid inequality as

$$\sum_{i\in\Omega:i'\in\widehat{V}\backslash\Omega} x_i\geq 1. \tag{8}$$

In the two versions of our B&C approach, we always implement the separation method to find a valid cut with minimal cardinality. At each round in the B&C1 approach, at most one cut is added to the node LP of the B&B tree and the LP subproblem is solved again. In the B&C2 approach, there may be more than one violated cut added to the node LP, at each round, depending on whether the cut returned by the separation method is an FDI (see Section 5.3.3). At each node of the B&B tree, we repeat the separation method until no violated cut is found.

5.3.3. Lifting non-facet-defining inequalities

In the B&C2 approach, we also implement the lifting procedure. Suppose the separation method returns a valid DSN $D=\{i\in\Omega: i'\in\widehat{V}\backslash\Omega\}$. The lifting procedure works as follows.

Following the proof of Theorem 2, we first create graph M_i for each node $i\in D$ and determine the set S_i of all cut-vertices of M_i. Let $C=\bigcap_{i\in D}S_i$. Then, we add the violated cut(s) in one of the following cases:

(1) Case 1: $C=\emptyset$.
In this case, we add an FDI to the node LP and resolve it again.
(2) Case 2: $C\neq\emptyset$.

In this case, we first get a valid lifted inequality for each node $j\in C$.

$$\sum_{i\in D}x_i+x_j\geq 2. \tag{9}$$

We then check if each inequality (9) is an FDI. For each node $j\in C$, graph $\widehat{M}_j$ is created and $\widehat{S}_j$ is determined. For each node $j\in C$, we then add cuts as follows:

If $p=|D\bigcap\widehat{S}_j|\leq 2$ and $q=|\{D\bigcup\{j\}\}\bigcap\widehat{S}_k|\leq 2$ for all $k\in C\backslash\{j\}$, we add to the node LP inequality (9), which is an FDI.

Otherwise, if $p>2$, we add a stronger lifted cut:

$$\sum_{i\in D}x_i+(p-1)x_j\geq p. \tag{10}$$

Else if $q=|\{D\bigcup\{j\}\}\bigcap\widehat{S}_k|>2$ for some $k\in N\backslash\{D,j\}$, we add a stronger lifted cut:

$$\sum_{i\in D}x_i+x_j+(q-1)x_k\geq q. \tag{11}$$

5.4. Pruning and speed-up strategies

We first present a pruning rule to manage the enumeration tree in our B&C approach. Let x_i^+ and x_i^- respectively be the upper and lower bound on variable x_i. Because of branching, lower or upper bound of x_i might change in the B&B tree. At node k of the B&B tree, define a set $\widetilde{V}$ of candidate regenerators, i.e., $\widetilde{V}=V\backslash\{i: x_i^+=0, i\in V\}$. Further, define a restricted graph $\widehat{G}_k=(\widehat{V}_k,\widehat{E}_k)$ with node set $\widehat{V}_k=\widetilde{V}$ and edge set $\widehat{E}_k$. Let Q_k be the set of cut-vertices of graph $\widehat{G}_k$. We present the pruning rule below.

Lemma 9. *For node k of the B&B tree, define set $\widetilde{V}$ of candidate regenerators. Node k can be removed from the search tree if the RLP instance is still infeasible with all nodes in $\widetilde{V}$ equipped with regenerators.*

Proof. The proof is straightforward. Set $\widetilde{V}$ defines the network nodes where we can place regenerators. Suppose all nodes in $\widetilde{V}$ are equipped with regenerators. If the instance becomes infeasible, it is not possible to find feasible solutions at any child node of the current node k in the B&B tree. □

If a node k cannot be pruned from the search tree, we may further fix some variables to one following Lemma 10.

Lemma 10. *Suppose node k cannot be removed from the B&B tree according to Lemma 9. If $Q_k\neq\emptyset$, then we can set the lower bound of x_i, $\forall i\in Q_k$, equal to one at node k of the B&B tree.*

Proof. Because node k cannot be removed from the enumeration tree according to Lemma 9, nodes in $\widehat{V}_k$ must be connected. Furthermore, if graph $\widehat{G}_k$ is not bi-connected, we must put regenerators at all of its cut-vertices. □

In our B&C approach, we implement the strategies in Lemmas 9 and 10, when depth of a node k in the B&B tree is greater than $0.35n$ and node k is created by down branching.

Consider an RLP instance. At any node k of the B&B tree, let UB and LB respectively be the objective value of the best returned integer solution and the best known bound of all of the remaining open nodes in the enumeration tree. Then the B&C approach optimally solves the RLP instance if at every node k of the B&B tree, $UB - LB < 1$. Our computational results demonstrate that this termination criteria can reduce the overall computational time for small instances.

5.5. Implementation strategy of the B&C approach

Chen et al. (2010) implemented their B&C approach using CPLEX 10.0. To compare our approach with that in Chen et al. (2010), we also used CPLEX 10.0 as a framework to implement our B&C approach. The LP solver in CPLEX 10.0 was called through CPLEX Callable Library C API in the B&C approach. We turn off all default cut generation routines in CPLEX. It means that we only add the cuts identified by our separation approach. We also deactivate the primal heuristic in CPLEX. We applied the best-bound rule to select a new active node in the B&B tree, i.e., the node with the best objective function for the associated LP relaxation is selected to process when backtracking. We left other parameters at their default values.

6. Computational evaluation

In this section, we investigate the performance of the proposed B&C approach on 400 RLP and 47 MCDSP/MLSTP benchmark instances. The two versions of our B&C approach were coded in C and compiled on VC++ 2010. All the experiments were implemented on a PC 2.0 gigahertz with 2 gigabyte RAM. In all of the experiments, the computational time is reported in seconds.

6.1. Test methods in the comparison study

We first compare our proposed approach with an existing exact method for the RLP in the literature. Since the RLP, MCDSP and MLSTP are equivalent problems, an optimal solution of the RLP also defines an optimal solution to the MCDSP and the MLSTP. Therefore, we further compare our B&C approach with the best exact algorithms for the MCDSP and the MLSTP. We list two versions of our B&C approach and ten available exact algorithms for the RLP, MCDSP and MLSTP:

- B&C1: our B&C approach without lifting procedure.
- B&C2: our B&C approach in which we lift non-facet-defining inequality to one stronger inequality, or a set of stronger inequalities.
- B&C _ CLR: the B&C approach for the RLP, proposed by Chen et al. (2010).
- DGR: the B&C algorithm for the MLSTP, developed by Lucena et al. (2010).
- MTZ: the B&B algorithm based on MTZ (Miller-Tucker-Zemlin) MCDSP reformulation suggested in Fan and Watson (2012).
- BCS: the B&C algorithm for the MCDSP, developed by Simonetti et al. (2011).
- SABE and IPBE: two variants of Benders decomposition algorithms for the MCDSP, developed by Gendron et al. (2014).
- SABC and IPBC, two variants of B&C approaches for the MCDSP, developed by Gendron et al. (2014).
- SAHY and IPHY, two variants of hybrid algorithms combing decomposition and B&C algorithms for the MCDSP, developed by Gendron et al. (2014).

We did not include method of Rahman (2012), Rahman et al. (2015) in our comparison study, because as mentioned earlier, this method is a preliminary implementation of the B&C algorithm, and its performance is not demonstrated by comparing it with best exact algorithms in the literature.

Chen et al. (2010) used CPLEX 10.0 as a framework to implement their B&C algorithm on a Pentium D 3.0 gigahertz PC with 2 gigabyte RAM. Lucena et al. (2010) coded their algorithm in C++ using MIP solver XPRESS 19.0 as the implementation framework and tested it on a 3.00 gigahertz Intel Xeon X5472 with 16 gigabyte RAM. A time limit of 3 hours was imposed on every single run. Simonetti et al. (2011) and Gendron et al. (2014) used MIP solver XPRESS 19.0 as the framework to implement their algorithms on a 2.0 gigahertz Intel Xeon X5405 with 8 gigabyte RAM. A time limit of 1 hour was set on every single run. Gendron et al. (2014) finally implemented XPRESS to solve the MTZ reformulation for the MCDSP in Fan and Watson (2012).

6.2. Test instances

As mentioned above, we will compare our B&C approaches with ten exact algorithms for the RLP, MCDSP and MLSTP. Our test instances include two parts: 400 RLP benchmark instances and 47 MLSTP/ MCDSP benchmark instances.

6.2.1. RLP benchmark instances

Chen et al. (2010) tested their heuristics and B&C approach on a set of 740 instances with three types of networks: randomly generated networks, networks with random distances and Euclidean networks. Chen, Ljubić, and Raghavan (2012) pointed out that instances with randomly generated networks were the most difficult to solve, which was also verified by the computational results reported in Chen et al. (2010). Currently only randomly generated instances can be downloaded from Prof. Ivana Ljubićs homepage (http://homepage.univie.ac.at/ivana.ljubic/research/rlp/). For the purpose of comparisons, we also tested our approaches on these instances with randomly generated networks. Chen et al. (2010) generated each instance according to two parameters: n and $p \in [0, 1]$, which denotes the percentage of NDC node pairs. In each instance, there are $\lceil p \times (\frac{n^2-3n+2}{2})\rceil$ NDC node pairs (Chen et al., 2010). These two parameters also control the computational difficulty of each instance. In total, Chen et al. (2010) generated two sets of 400 instances with $p = 10$ percent, 30 percent, 50 percent, 70 percent, and 90 percent. Set 1 consists of 200 small instances with up to 100 nodes. Chen et al. (2010) generated 200 additional instances with 200–500 nodes in set 2 to investigate the performance of their methods on large instances. As in Chen et al. (2010), we deleted “percent” for p while reporting experimental results.

With these 400 benchmark instances, we report on our experience with the proposed B&C approaches. As in Chen et al. (2010), we also set a run time limit of 3600 seconds for each instance. In the following tables, the results of Chen’s methods are directly taken from Chen et al. (2010), given that a similar computational environment was used here.

6.2.2. MLSTP/MCDSP benchmark instances

Using 47 MLSTP/MCDSP instances as the test bed, Gendron et al. (2014) compared their six algorithms with three other best exact algorithms for the MLSTP and MCDSP. We also used these 47 instances as benchmark instances (set 3). The first 41 instances in set 3 are MLSTP instances introduced in Lucena et al. (2010). Each of these instances is labelled as n_ dx with number of nodes $n \in \{30, 50, 70, 100, 120, 150, 200\}$ and network density x ∈ {5 percent, 10 percent, 20 percent, 30 percent, 50 percent, 70 percent}. The other five instances in set 3 are IEEE reliability test instances. Fan and Watson (2012) generated these five instances with number

Table 1
Computational results under different threshold levels.

n	p	$\xi = 0$		$\xi = 0.2$		$\xi = 0.4$		$\xi = 0.6$		$\xi = 0.8$		$\xi = 1$		Best solution	
		NR	#Opt	NR	#Opt	NR	#Opt	NR	#Opt	NR	#Opt	NR	#Opt	NR	#Opt
40	10	1.4	10	1.4	10	1.4	10	1.4	10	1.4	10	1.4	10	1.4	10
	30	2	10	2	10	2	10	2	10	2	10	2	10	2	10
	50	3.1	9	3.1	9	3	10	3	10	3	10	3	10	3	10
	70	4.8	5	4.8	5	4.7	6	4.6	7	4.6	7	4.6	7	4.6	7
	90	9.8	3	10	2	10	3	10.1	1	9.8	3	9.6	4	9.4	6
60	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10
	30	2	10	2	10	2	10	2	10	2	10	2	10	2	10
	50	3.1	9	3.1	9	3.2	8	3.2	8	3.2	8	3.2	8	3.1	9
	70	5.1	7	5.1	7	5.1	7	5.1	7	5.1	7	5.1	7	5	8
	90	11.7	1	11.7	1	11.5	1	11.3	2	11	4	11	4	11	4
80	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10	1.9	10
	30	2.7	6	2.7	6	2.7	6	2.7	6	2.7	6	2.7	6	2.7	6
	50	3.5	7	3.5	7	3.5	7	3.5	7	3.5	7	3.5	7	3.5	7
	70	5.8	2	5.8	2	5.8	2	5.8	2	5.8	2	5.8	2	5.8	2
	90	12.6	1	12.5	1	12.6	1	12.4	1	12.3	1	12.3	2	12.2	2
100	10	2	10	2	10	2	10	2	10	2	10	2	10	2	10
	30	2.8	9	2.8	9	2.8	9	2.8	9	2.8	9	2.8	9	2.8	9
	50	4	9	4	9	4	9	4	9	4	9	4	9	4	9
	70	5.9	3	5.9	3	5.9	3	5.9	3	5.9	3	5.9	3	5.9	3
	90	13.3	2	13.2	1	13.1	1	13.2	0	13	0	12.8	0	12.6	2
Total		99.4	133	99.4	131	99.1	133	98.8	132	97.9	136	97.5	138	96.8	144

of nodes $n \in \{14, 30, 57, 73, 118, 300\}$ and network density ranging from less than 1 percent to nearly 15 percent.

As in Gendron et al. (2014), we also set a run time limit of 3600 seconds for each instance. The computational results of the other algorithms for these 47 instances are directly taken from Gendron et al. (2014).

6.3. Performance of Procedure 1

In Section 5.2, we design a threshold based procedure to produce initial integer solutions in the B&C approach. We first examined how the threshold levels affect the performance of this simple procedure. Using 200 RLP instances, we implemented Procedure 1 under six threshold levels. Table 1 summarizes the average computational results for each threshold level. The last two columns indicate the best result for each instance among the six threshold levels. For each pair of n and p, we present two performance statistics:

- NR: the average number of regenerators returned by the procedure.
- #Opt: the number of instances, out of 10 instances, that were optimally solved within the time limit.

Computational results in Table 1 show that there is a slight difference in solution quality among six threshold levels. On average, Procedure 1 worked better with larger threshold level, in terms of the number of instances optimally solved. But with larger threshold level, it did not necessarily find the best solution for each instance. For example, 138 instances were optimally solved by Procedure 1 with threshold $\xi = 1$. But, as one may observe from the last two columns in Table 1, Procedure 1 with the best threshold combination, optimally solved 144 instances.

Recall that Procedure 1 can also be implemented as a simple heuristic for the RLP. To demonstrate its performance, we implemented Procedure 1 to solve 320 instances and compared it with three heuristics (Greedy, H1 and H2) developed by Chen et al. (2010) and the new implementation of these three heuristics (Greedy*, H1* and H2*) by Duarte et al. (2014). The average computational results are shown in Table 2. Column Avg. represents the average running time required to solve ten instances in each group. For our Procedure 1, the Avg. is the sum of computational times for the six threshold levels. For each instance group, the method that got minimum average upper bounds, is marked in bold.

The computational results in Table 2 show that Procedure 1 can be used as a good heuristic for the RLP. We first focus on comparing Procedure 1 and Greedy, H1 and H2 in Chen et al. (2010). Of the 200 instances, Greedy, H1 and H2 respectively found optimal solutions for 116, 104 and 106 instances. Our heuristic obtained the optimal solution for 144 instances. Among the other 120 large instances with 200, 300, 400, and 500 nodes, the H2 heuristic found optimal solutions for 35 instances, whereas our heuristic found optimal solutions for 68 instances. We then compare Procedure 1 and Greedy*, H1* and H2*. As one may observe, no method outperformed the other three methods on all instances. The computational results also show that Procedure 1 is competitive with Greedy*, H1* and H2*. Considering these 200 instances, Procedure 1 found slightly better results, in term of the average number of regenerators required.

As an indicative comparison, we also compared Procedure 1 with H2*, GRASP and BRKGA developed by Duarte et al. (2014). A total of 280 instances including the 200 instances in set 1 are used as the test bed in Duarte et al. (2014). Note that this comparison study includes 80 instances with $n \in \{40, 60, 80, 100\}$ and $p \in \{20, 40, 60, 80\}$, which were not used in Chen et al. (2010). We thus did not use them for evaluating our B&C approach. Procedure 1 was also used to solve these instances. The comparison results are presented in Table 3. Each row represents the average results for 70 instances with each network size. The results show that Procedure 1 always outperformed H2*. Procedure 1 only found better upper bounds than BRKGA on instances with $n = 60$, whereas BRKGA found better results on instances with $n = 40$ and $n = 80$. GRASP always found better results than Procedure 1.

To summarize, our comparison results show that although it is not our main contribution, Procedure 1 reaches a good performance.

6.4. Impact of lifted cuts

In Section 3, we presented the necessary and sufficient conditions for a minimal DSN to correspond to an FDI, and how to lift a non-facet-defining inequality to get one or more stronger lifted-inequalities. We now attempt to evaluate the effects of lifted

Table 2
Computational evaluation of Procedure 1 (part one).

n	*p*	Greedy			Greedy*		H1			H1*		H2			H2*		Procedure 1		
		NR	Avg.	#Opt	NR	Avg.	NR	Avg.	#Opt	NR	Avg.	NR	Avg.	#Opt	NR	Avg.	NR	Avg.	#Opt
40	10	**1.4**	0.00	10	**1.4**	0.00	**1.4**	0.00	10	**1.4**	0.00	**1.4**	0.10	10	**1.4**	0.00	**1.4**	0.00	10
	30	**2.0**	0.20	10	**2.0**	0.01	**2.0**	0.10	10	**2.0**	0.00	**2.0**	0.00	10	**2.0**	0.00	**2.0**	0.00	10
	50	**3.0**	0.10	10	**3.0**	0.01	3.1	0.30	9	**3.0**	0.00	3.3	0.10	7	**3.0**	0.00	**3.0**	0.00	10
	70	4.6	0.70	7	4.5	0.01	4.7	0.50	6	4.6	0.00	4.6	0.20	7	**4.4**	0.00	4.6	0.00	7
	90	9.5	1.70	5	10.1	0.02	9.7	1.30	2	9.9	0.01	9.8	0.50	2	10.2	0.00	**9.4**	0.00	6
60	10	**1.9**	0.30	10	**1.9**	0.04	**1.9**	0.30	10	**1.9**	0.01	**1.9**	0.00	10	**1.9**	0.00	**1.9**	0.00	10
	30	2.1	0.80	9	**2.0**	0.03	2.1	0.30	9	**2.0**	0.01	2.2	0.20	8	**2.0**	0.00	**2.0**	0.00	10
	50	3.2	1.90	8	3.1	0.05	3.3	1.10	7	3.1	0.00	3.4	0.40	6	**3.0**	0.00	3.1	0.01	9
	70	5.3	4.30	5	5.1	0.06	5.4	3.90	4	5.1	0.00	5.3	0.70	5	**5.0**	0.01	**5.0**	0.01	8
	90	**10.8**	15.80	4	11.0	0.10	11.2	18.00	2	11.4	0.01	11.2	2.50	2	11.3	0.01	11.0	0.00	4
80	10	**1.9**	1.30	10	**1.9**	0.06	**1.9**	0.70	10	**1.9**	0.03	**1.9**	0.10	10	**1.9**	0.00	**1.9**	0.01	10
	30	2.9	3.90	4	**2.6**	0.09	2.8	2.00	5	2.7	0.02	2.8	0.30	5	2.7	0.01	2.7	0.01	6
	50	3.8	9.10	2	**3.5**	0.08	4.0	5.60	0	3.6	0.01	3.7	0.90	3	**3.5**	0.00	**3.5**	0.02	7
	70	6.0	19.40	0	5.8	0.13	5.9	17.30	1	**5.7**	0.00	5.8	3.00	2	**5.7**	0.01	5.8	0.01	2
	90	**11.9**	88.90	3	12.2	0.22	12.5	70.70	1	12.4	0.03	12.8	13.00	0	12.1	0.03	12.2	0.01	2
100	10	**2.0**	3.60	10	**2.0**	0.14	**2.0**	1.80	10	**2.0**	0.07	**2.0**	0.50	10	**2.0**	0.01	**2.0**	0.02	10
	30	2.9	9.60	8	**2.7**	0.17	2.9	7.10	8	2.8	0.03	2.8	1.40	9	2.8	0.01	2.8	0.02	9
	50	**4.0**	21.10	0	**4.0**	0.20	**4.0**	13.60	0	**4.0**	0.02	**4.0**	2.60	0	**4.0**	0.01	**4.0**	0.04	9
	70	6.0	49.00	1	6.0	0.25	6.4	56.00	0	**5.8**	0.01	6.1	8.20	0	6.0	0.01	5.9	0.03	3
	90	13.4	286.40	0	12.7	0.38	13.7	295.00	0	13.5	0.04	13.3	42.90	0	13.0	0.02	**12.6**	0.02	2
200	10											**2.0**	4.70	10			**2.0**	0.14	10
	30											**3.0**	22.90	0			**3.0**	0.23	10
	50											4.9	65.20	0			**4.2**	0.34	8
	70											7.4	212.80	0			**6.9**	0.26	0
300	10											**2.0**	24.60	10			**2.0**	0.23	10
	30											3.7	140.90	0			**3.0**	0.48	9
	50											**4.9**	459.40	0			**4.9**	0.83	0
400	10											**2.0**	101.70	10			**2.0**	1.20	10
	30											3.9	514.50	0			**3.1**	1.03	1
	50											5.5	1611.30	0			**5.0**	0.86	0
500	10											**2.0**	210.50	5			**2.0**	1.14	10
	30											4.0	1221.70	0			**3.8**	1.98	0

Note: Duarte et al. (2014) coded Greedy*, H1* and H2* in Java and implemented them on a Pentium 4, 3.0 gigahertz with 2 gigabyte RAM.

Table 3
Computational evaluation of Procedure 1 (part two).

n	H2*		GRASP		BRKGA		Procedure 1	
	NR	Avg.	NR	Avg.	NR	Avg.	NR	Avg.
40	3.96	0.003	3.77	0.25	3.83	1.44	3.84	0.0015
60	4.37	0.006	4.26	0.65	4.34	4.46	4.31	0.0046
80	4.90	0.01	4.50	1.36	4.67	9.74	4.81	0.0121
100	5.27	0.02	4.91	2.39	5.00	20.17	5.00	0.0259

Note: Duarte et al. (2014) coded GRASP and BRKGA in Java and tested them on a Pentium 4, 3.0 gigahertz with 2 gigabyte RAM.

cuts on the overall performance of our approach. We ran B&C1 and B&C2 to solve 120 RLP instances with large density (e.g., they are more difficult to solve). The average computational results are shown in Table 4. We introduce seven new performance measures:

- LB: The best known bound of all of the remaining open nodes in a B&B tree at termination, which is the best known lower bound on the optimal solution value of each instance. When an instance has been solved to optimality, the value of LB matches the optimal solution value.
- Rgap: The relative objective gap between the best node value (LB) and the objective value (UB) of the incumbent solution at termination. The Rgap is computed as $\frac{\text{UB -LB}}{\text{UB}} \times 100$.
- #NoSep: the number of instances that were optimally solved without separation.
- #NoCut: the number of instances for which no violated cuts were identified from the separation. This performance measure only applies to instances, for which separation occurs during the solution process.
- #Impt: the average number of total separation problems solved.
- %Success: the average success rate of violated cuts that can be identified among the solved separation problems.
- %Lift: the average success rate of identified inequalities not being FDIs.

The computational results in Table 4 show that B&C1 managed to solve 101 instances. B&C2 obtained a higher success rate. In addition to these 101 instances, B&C2 also solved seven other instances. Considering the instances solved to optimality, B&C2 produced much better results, in terms of average CPU time. For those instances that were not solved by the two algorithms, B&C2 always terminated with better upper and lower bounds. B&C2 outperformed B&C1 due to the fact that lifted stronger cuts speeded up the algorithm. The strength of B&C2 will be further verified by experiments in the following sections.

Further, the computational results verify the impact of the lifted cuts. Over instances with $p = 70$ and $n \in \{40, 60, 80\}$, most of the cuts found by the separation problem were FDIs. As network density increased, the separation method started to find non-facet-defining cuts. As we can observe from our results, B&C2 did benefit

Table 4
Effects of lifted cuts.

n	p	B&C1				B&C2								
		NR	Rgap	Avg.	#Opt	NR	Rgap	Avg.	#Opt	#NoSep	#Impt	#NoCut	%Success	%Lift
40	70	4.3	0	0.05	10	4.3	0	0.06	10	4	5.7	0	73.13	0.00
	90	8.9	0	0.08	10	8.9	0	0.10	10	1	72.5	1	76.11	24.45
60	70	4.8	0	0.31	10	4.8	0	0.42	10	0	41.1	1	56.48	0.00
	90	10.2	0	0.39	10	10.2	0	0.18	10	0	137.8	0	86.18	12.75
80	70	5	0	1.76	10	5	0	1.86	10	0	133.3	3	51.83	0.00
	90	11	0	44.41	10	11	0	1.90	10	0	1618.2	0	91.56	11.16
100	50	3.9	0	4.29	10	3.9	0	3.20	10	0	226.9	0	38.47	5.31
	70	5.2	0	36.78	10	5.2	0	7.31	10	0	585.4	0	70.43	8.42
	90	11.8	0	166.42	10	11.8	0	13.61	10	0	3459.8	0	91.81	16.35
200	50	4	0	146.42	10	4	0	156.96	10	0	1338.4	0	70.63	6.53
	70	6.9	32.41	3422.88	1	6.4	14.35	1679.71	6	0	62001.1	0	82.83	12.91
	90	14.9	24.42	3600	0	14.8	17.01	3461.43	2	0	127633	0	90.05	15.20

Table 5
Comparison results on RLP instances in set 1.

n	p	B&C _ CLR			B&C1					B&C2				
		LB	Avg.	#Opt	NR	LB	Rgap	Avg.	#Opt	NR	LB	Rgap	Avg.	#Opt
40	10	1.4	0.8	10	1.4	1.4	0	0.02	10	1.4	1.4	0	0.02	10
	30	2.0	3.1	10	2.0	2.0	0	0.02	10	2.0	2.0	0	0.02	10
	50	3.0	4.6	10	3.0	3.0	0	0.05	10	3.0	3.0	0	0.04	10
	70	4.3	18.9	10	4.3	4.3	0	0.05	10	4.3	4.3	0	0.06	10
	90	8.9	3.5	10	8.9	8.9	0	0.08	10	8.9	8.9	0	0.10	10
60	10	1.9	7.0	10	1.9	1.9	0	0.02	10	1.9	1.9	0	0.02	10
	30	2.0	8.2	10	2.0	2.0	0	0.04	10	2.0	2.0	0	0.03	10
	50	3.0	48.9	10	3.0	3.0	0	0.08	10	3.0	3.0	0	0.06	10
	70	4.8	252.1	10	4.8	4.8	0	0.31	10	4.8	4.8	0	0.42	10
	90	10.2	37.0	10	10.2	10.2	0	0.39	10	10.2	10.2	0	0.18	10
80	10	1.9	19.3	10	1.9	1.9	0	0.02	10	1.9	1.9	0	0.03	10
	30	2.3	201.9	10	2.3	2.3	0	0.13	10	2.3	2.3	0	0.11	10
	50	2.9	1834.6	6	3.2	3.2	0	0.51	10	3.2	3.2	0	0.49	10
	70	4.9	985.5	9	5.0	5.0	0	1.76	10	5.0	5.0	0	1.86	10
	90	10.7	1543.0	7	11.0	11.0	0	44.41	10	11.0	11.0	0	1.90	10
100	10	2.0	124.5	10	2.0	2.0	0	0.03	10	2.0	2.0	0	0.04	10
	30	2.7	1024.9	10	2.7	2.7	0	0.26	10	2.7	2.7	0	0.26	10
	50	2.4	3439.1	1	3.9	3.9	0	4.29	10	3.9	3.9	0	3.20	10
	70	4.2	3211.8	2	5.2	5.2	0	36.78	10	5.2	5.2	0	7.31	10
	90	10.0	3600.0	0	11.8	11.8	0	166.42	10	11.8	11.8	0	13.61	10
Average		4.28	818.44	8.25	4.53	4.53	0	12.78	10	4.53	4.53	0	1.49	10

from lifted stronger cuts, which led to higher lower bounds, lower upper bounds and reduced overall computational time.

6.5. Comparison results on RLP instances in set 1

On 200 RLP instances with up to 100 nodes, we next evaluate performance of our B&C approaches by comparing it with B&C _ CLR developed by Chen et al. (2010). Table 5 summarizes the comparison results. A Rgap of zero indicates the B&C approach solved an instance to optimality. Since the RLP is very difficult to solve, Chen et al. (2010) only reported the best lower bounds obtained by their B&C method at termination and asserted the heuristic found an optimal solution if the heuristic solution equals to the lower bound. If the time limit of 3600 seconds was exceeded and the instance was not optimally solved, the computational time was recorded as 3600 seconds.

The computational results show that B&C _ CLR produced optimality certificates for 165 of the 200 test instances. Our two approaches B&C1 and B&C2 both solved all of these 200 instances. As one may observe from the results, B&C1 and B&C2 worked much faster than B&C _ CLR, especially for instances with larger densities. Additionally, considering instances solved to optimality, B&C2 produced much better results than B&C1, in terms of average running time. On average, B&C1 consumed 12.78 seconds to solve each instance. However, B&C2 required only 1.49 seconds to produce optimality certificate for each instance (see the last column of Table 5).

6.6. Comparison results on RLP instances in set 2

In this section, we further verify the performance of our B&C approaches on large scale RLP instances in set 2. Table 6 gives the average computational results of our approaches and B&C _ CLR. Each entry of " – " indicates that Chen et al. (2010) did not run their B&C approach to solve these instances.

The computational results show that within the time limit, B&C _ CLR could solve 20 instances of the 200 test instances. As one may observe from the results in Table 6, B&C _ CLR respectively solved ten 200- and 300-node instances with low density (e.g., $p=10$ percent). On these 20 instances, B&C1 and B&C2 were much faster than B&C _ CLR.

For the instances that were not solved by B&C _ CLR, B&C1 and B&C2 always obtained a better, or at least equal, lower bound. B&C1 and B&C2 significantly extended the ability to solve large RLP instances. In addition to these 20 instances solved by B&C _ CLR, B&C1 and B&C2 also managed to solve 69 and 87 instances, respectively. Once again, B&C2 always produced better results than B&C1 in terms of running time and lower and upper bounds at termination. Considering the instances solved to optimality by the two approaches, B&C2 always required less running time. For those

Table 6
Comparison results on large RLP instances in set 2.

n	p	B&C _ CLR			B&C1					B&C2				
		LB	Avg.	#Opt	NR	LB	Rgap	Avg.	#Opt	NR	LB	Rgap	Avg.	#Opt
200	10	2.0	331.60	10	2.0	2.0	0.00	0.17	10	2.0	2.0	0.00	0.15	10
	30	2.0	3600	0	3.0	3.0	0.00	36.44	10	3.0	3.0	0.00	12.46	10
	50	2.0	3600	0	4.0	4.0	0.00	146.42	10	4.0	4.0	0.00	156.96	10
	70	4.0	3600	0	6.9	4.6	32.41	3422.88	1	6.4	5.4	14.35	1679.71	6
	90	–	–	–	14.9	11.3	24.42	3600	0	14.8	12.2	17.01	3461.43	2
300	10	2.0	1765.90	10	2.0	2.0	0.00	0.55	10	2.0	2.0	0.00	0.54	10
	30	2.0	3600	0	3.0	3.0	0.00	163.69	10	3.0	3.0	0.00	36.02	10
	50	2.0	3600	0	4.8	3.1	33.25	3260.61	2	4.2	3.8	8.46	1372.27	8
	70	–	–	–	7.1	4.3	39.48	3600	0	7.0	4.8	31.54	3600	0
	90	–	–	–	17.2	11.0	35.81	3600	0	16.8	11.9	31.20	3600	0
400	10	2.0	3600	0	2.0	2.0	0.00	1.29	10	2.0	2.0	0.00	1.29	10
	30	2.0	3600	0	3.0	3.0	0.00	1277.96	10	3.0	3.0	0.00	201.21	10
	50	2.0	3600	0	5.0	2.6	48.57	3600	0	4.9	3.0	40.91	3386.47	1
	70	–	–	–	7.9	4.0	48.82	3600	0	7.9	4.4	45.31	3600	0
	90	–	–	–	18.9	10.7	43.60	3600	0	18.9	11.5	39.03	3600	0
500	10	1.5	3600	0	2.0	2.0	0.00	2.56	10	2.0	2.0	0.00	2.04	10
	30	2.0	3600	0	3.2	2.4	22.86	2631.26	6	3.0	3.0	0.00	539.82	10
	50	–	–	–	5.0	2.4	51.43	3600	0	5.0	2.9	41.39	3600	0
	70	–	–	–	8.3	3.7	55.04	3600	0	8.2	4.3	47.54	3600	0
	90	–	–	–	20.4	10.5	48.66	3600	0	20.2	11.2	44.46	3600	0
Average		2.13	3174.79	1.7	7.0	4.6	24.22	2167.23	4.45	6.9	5.0	18.06	1802.54	5.35

instances that were not solved by our two approaches within the time limit, B&C2 always found better, or at least equal, integer solutions and lower bounds (this is also indicated by column Rgap). This is explained by the strength of the lifting procedure in B&C2. The lifted inequalities speeded up the solution process.

To summarize, the computational results in Tables 5 and 6 show that the two versions of our B&C approach are efficient algorithms for the RLP. Due to strength of lifted inequalities, B&C2 always found significantly better results than B&C1.

6.7. Comparison results on MLSTP/MCDSP instances in set 3

We finally focus on comparing B&C1 and B&C2 with nine other exact algorithms for the MLSTP/MCDSP. The 47 benchmark instances were used in this comparison study. Table 7 summarizes the computational results of B&C1, B&C2 and the other nine algorithms. In Table 7, notation “–” shows that the time limit of 3600 seconds was exceeded and the instance was not optimally solved. Table 8 presents the performance difference between our approaches and other algorithms. We introduce the following performance statistics:

- CPU: The computational time required by each algorithm to solve each instance.
- $\triangle$OPT: The difference between the number of instances optimally solved by our approach, compared to each of the other algorithms. A positive $\triangle$OPT means that our approach found optimal solutions for more instances.
- Min, Max and Average: We compute difference between the total computational time required by our approach and each of the other algorithm to solve the instances that the two algorithms can solve to optimality. The Min, Max, and Average denote the minimum, maximum, and average value of CPU difference across all the instances.
- $\triangle$CPU: The total CPU difference across all the instances. A negative $\triangle$CPU indicates our approach was faster.

The computational results in Table 7 show that the branch-and-bound algorithm based on the MTZ model is not competitive with the other approaches found in the literature, i.e., it only solved 35 of the 47 instances. Therefore, we do not discuss the comparison between our approaches and MTZ in the following section. The two versions of our B&C approach attained the highest success rates among the algorithms considered in this comparison study. They both solved 45 of the 47 instances within the time limit.

We first focus on comparing our B&C approaches and four available branch-and-cut algorithms. In terms of optimality certificates provided by five B&C approaches, DGR is the best B&C algorithm in the literature, which solved 40 instances to optimality. Both B&C1 and B&C2 outperform DGR. On the same set of instances for which our B&C approaches and DGR provided optimality certificates, B&C1 and B&C2 required significantly less computational time. Our two approaches also worked better than two the B&C algorithms (SABC and IPBC) in Gendron et al. (2014). SABC and IPBC respectively found optimal solutions in 38 and 36 instances. Column with heading $\triangle$CPU demonstrates that our two approaches were faster than SABC and IPBC, especially for large-scale instances. B&C1 and B&C2 both found optimal solutions for more instances than BCS.

We next compare our two approaches with Benders decomposition algorithms in Gendron et al. (2014). SABE and IPBE respectively obtained optimal solutions only in 37 and 39 out of the 47 instances. Gendron et al. (2014) have shown that SABE and IPBE usually performed well if just a few feasibility cuts are required to attain optimality, and IPBE is usually faster than SABE. Among nine exact algorithms found in the literature, IPBE is the only one which could solve instance 200_d5. Our approaches solved this instance in less time.

We finally compare our approaches with SAHY and IPHY. As Gendron et al. (2014) demonstrated, their hybrid algorithms usually performed better than other algorithms. SAHY and IPHY managed to solve 40 and 41 instances, respectively. Our two approaches have a better overall success rate. Additionally, considering instances solved to optimality by hybrid algorithms, our approaches saved total computational time.

We have to point out that our work and Gendron et al. (2014), respectively, used CPLEX and XPRESS as the implementation platform, although we carried out experiments on a PC which has similar computational speed to the one used in Gendron et al. (2014). In summary, our experimental results show that the two versions of our B&C approach are both competitive with other exact algorithms in the literature, and obtained better overall success rate. Our approaches solved more instances to optimality with less running time.

Table 7
Comparison results on MLSTP/MCDSP instances in set 3.

Instance	DGR	BCS	SABC		IPBC		SABE		IPBE		SAHY		IPHY		MTZ		B&C1		B&C2	
	CPU	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU
30_ d10	0.01	0.01	15	0.03	15	0.02	15	1222.10	15	756.87	15	6.11	15	2.88	15	36.20	15	1.64	15	0.20
30_ d20	0.10	0.02	7	0.02	7	0.02	7	0.01	7	0.00	7	0.01	7	0.02	7	0.92	7	0.05	7	0.02
30_ d30	0.03	0.05	4	0.05	4	0.06	4	0.02	4	0.02	4	0.03	4	0.02	4	1.23	4	0.02	4	0.00
30_ d50	0.08	0.04	3	0.01	3	0.03	3	0.00	3	0.00	3	0.01	3	0.01	3	1.07	3	0.02	3	0.02
30_ d70	0.01	0.02	2	0.02	2	0.00	2	0.00	2	0.00	2	0.01	2	0.00	2	0.07	2	0.02	2	0.00
50_ d5	0.01	0.02	31	0.01	31	0.02	31	–	31	–	31	88.63	31	9.46	31	182.06	31	3.84	31	0.02
50_ d10	0.36	0.42	12	0.82	12	0.20	12	34.30	12	3.09	12	2.69	12	3.89	12	5.18	12	0.19	12	0.17
50_ d20	1.32	0.66	7	0.77	7	0.97	7	0.21	7	0.09	7	0.40	7	0.16	7	4.23	7	0.08	7	0.08
50_ d30	1.21	0.25	5	0.32	5	0.25	5	0.18	5	0.11	5	0.45	5	0.31	5	14.90	5	0.05	5	0.05
50_ d50	0.51	0.25	3	0.23	3	0.06	3	0.00	3	0.01	3	0.01	3	0.01	3	4.80	3	0.02	3	0.02
50_ d70	0.04	0.29	2	0.24	2	0.01	2	0.00	2	0.00	2	0.01	2	0.01	2	0.74	2	0.02	2	0.02
70_ d5	0.26	1.42	27	2.06	27	0.39	29	–	29	–	27	188.65	27	674.75	27	2098.18	27	25.13	27	6.44
70_ d10	4.73	34.29	13	18.68	13	5.25	13	1.06	13	2.17	13	25.16	13	1.26	13	18.03	13	0.14	13	0.14
70_ d20	16.30	2.16	7	2.68	7	1.88	7	0.38	7	0.17	7	1.15	7	0.58	7	72.49	7	0.12	7	0.14
70_ d30	2.90	1.00	5	1.20	5	0.99	5	0.54	5	0.21	5	0.82	5	0.37	5	54.98	5	0.08	5	0.08
70_ d50	1.33	0.70	3	0.64	3	0.40	3	0.01	3	0.02	3	0.02	3	0.02	3	5.64	3	0.03	3	0.03
70_ d70	1.92	0.79	2	0.99	2	0.04	2	0.00	2	0.01	2	0.02	2	0.02	2	15.27	2	0.11	2	0.08
100_ d5	12.50	342.25	24	58.77	24	64.13	25	–	25	–	25	–	24	142.49	24	872.93	24	25.83	24	9.33
100_ d10	9.36	32.11	13	28.25	13	39.71	13	0.49	13	0.33	13	2.68	13	1.70	13	176.67	13	0.48	13	0.45
100_ d20	86.16	174.93	8	283.23	8	414.49	8	1.88	8	1.26	8	6.48	8	2.70	8	460.49	8	2.14	8	2.20
100_ d30	258.15	193.65	6	329.05	6	638.89	6	3.83	6	2.46	6	11.17	6	4.42	6	1462.83	6	2.47	6	2.39
100_ d50	132.55	35.41	4	48.00	4	41.51	4	1.55	4	0.76	4	3.23	4	1.56	4	101.29	4	1.05	4	0.45
100_ d70	154.10	12.03	3	13.20	3	12.02	3	1.55	3	0.03	3	1.57	3	0.91	3	50.68	3	0.30	3	0.20
120_ d5	2.65	-	25	1465.05	25	199.01	25	3.36	25	18.16	25	102.61	25	35.10	25	258.56	25	1.22	25	1.22
120_ d10	65.49	-	15	-	13	–	13	23.97	13	3.86	13	56.31	13	18.68	13	178.19	13	0.98	13	1.03
120_ d20	393.47	610.89	8	1316.70	8	–	8	5.02	8	3.79	8	16.47	8	8.31	8	1967.54	8	9.14	8	4.00
120_ d30	653.70	475.54	6	790.91	6	1913.36	6	5.25	6	4.44	6	14.21	6	7.56	6	2241.50	6	6.07	6	4.05
120_ d50	815.64	168.55	4	246.93	4	202.30	4	4.21	4	2.52	4	8.73	4	4.57	4	145.95	4	3.10	4	1.96
120_ d70	356.31	31.67	3	36.84	3	28.90	3	2.25	3	0.04	3	2.82	3	2.22	3	80.97	3	0.47	3	0.41
150_ d5	2954.00	-	27	-	27	–	27	–	26	771.07	27	–	26	–	26	-	26	173.34	26	117.67
150_ d10	3247.89	-	15	-	15	–	14	51.09	14	28.28	14	652.35	14	195.76	15	-	14	40.08	14	27.13
150_ d20	-	-	9	-	9	–	9	367.30	9	271.65	9	2116.24	9	903.76	9	-	9	219.09	9	153.69
150_ d30	2317.35	1954.00	6	2972.83	6	–	6	21.12	6	11.25	6	34.61	6	24.77	7	-	6	17.89	6	10.11
150_ d50	2756.36	481.61	4	724.92	4	477.10	4	7.81	4	5.78	4	17.57	4	10.79	4	257.47	4	5.19	4	5.12
150_ d70	1828.86	43.75	3	62.56	3	49.69	3	4.30	3	0.06	3	5.01	3	2.95	3	133.09	3	0.97	3	0.83
200_ d5	-	-	29	-	29	–	29	–	27	1658.85	29	–	28	–	29	-	27	1303.59	27	829.63
200_ d10	-	-	16	-	16	–	16	–	16	–	16	–	16	–	19	-	16	-	16	-
200_ d20	-	-	9	-	9	–	9	1686.30	9	1945.80	9	–	9	–	11	-	9	993.95	9	553.95
200_ d30	-	-	7	-	7	–	7	3210.83	7	1847.88	7	–	7	–	7	-	7	1094.05	7	579.89
200_ d50	-	2249.43	4	3363.33	4	1887.43	4	24.79	4	19.33	4	44.42	4	28.54	4	3509.21	4	22.01	4	10.55
200_ d70	-	271.91	3	340.20	3	275.84	3	10.53	3	0.13	3	9.17	3	5.63	3	507.44	3	6.76	3	4.79
14-Bus	0.01	0.00	5	0.01	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.20	5	0.02	5	0.00
30-Bus	0.01	0.01	11	0.01	11	0.01	11	0.00	11	0.00	11	0.00	11	0.01	11	0.52	11	0.05	11	0.02
57-Bus	0.06	0.00	31	0.02	31	0.03	31	–	31	–	31	24.40	31	192.43	31	-	31	23.82	31	5.93
RTS96	1.66	0.35	32	2.56	32	4.22	34	–	34	–	32	118.53	32	26.56	32	-	32	24.10	32	2.55
118-Bus	0.36	0.07	43	0.22	43	0.51	43	–	43	–	43	65.65	43	2.88	43	-	43	15.03	43	0.02
300-Bus	1076.89	-	139	-	136	–	139	–	139	–	139	–	138	–	∞	-	131	-	131	-

Note: DGR was implemented with a time limit of three hours. ∞ indicates the algorithm does not find a feasible integer solution within the time limit.

Table 8
Statistical results of B&C1, B&C2 and other algorithms.

Other algorithms	B&C1					B&C2				
	ΔOPT	ΔCPU	Min	Max	Average	ΔOPT	ΔCPU	Min	Max	Average
DGR	5	−15692.52	−3207.81	24.87	−402.37	5	−15873.24	−3220.76	6.18	−407.01
BCS	8	−6922.16	−2227.42	23.82	−187.09	8	−7047.74	−2238.88	5.93	−190.48
SABC	7	−11912.75	−3207.81	23.80	−313.49	7	−12038.33	−3352.78	5.91	−316.80
IPBC	9	−6087.16	−1907.29	24.74	−169.09	9	−6199.82	−1909.31	6.05	−172.22
SABE	8	−4266.23	−2116.78	4.12	−115.30	8	−5330.80	−2630.95	0.32	−144.08
IPBE	6	−3453.56	−951.85	11.80	−88.55	6	−5047.76	−1391.85	4.66	−129.43
SAHY	5	−3194.49	−1897.15	0.09	−79.86	5	−3381.86	−1962.55	0.06	−84.55
IPHY	4	−1858.31	−684.67	12.15	−45.32	4	−2062.19	−750.07	0.06	−50.30
MTZ	10	−14801.77	−3487.20	−0.06	−422.91	10	−14865.06	−3498.66	−0.07	−424.72

7. Summary and conclusions

In this paper, we studied the regenerator location problem in optical networks. We used a set covering formulation and studied the polyhedral structure of the convex hull of feasible solutions to the resulting set covering model. We derived necessary and sufficient conditions under which those constraints are FDIs and discussed how to lift a non-facet-defining inequality to get one or more stronger lifted inequalities. We further presented necessary and sufficient conditions under which a lifted inequality is an FDI. If the lifted inequality is not an FDI, we showed that it can be further lifted to obtain one more stronger lifted inequalities. With the proposed polyhedral results, we developed a new B&C approach to solve the RLP to optimality. Our B&C approach includes a simple but efficient procedure for generating starting incumbents, an efficient procedure for solving the separation problems to identify violated inequalities, lifting procedures, as well as pruning and speed-up strategies.

Over 447 benchmark instances, we carried out a series of computational experiments to evaluate the performance of two versions of our proposed B&C approach. With 400 RLP benchmark instances, we first compared our approaches with other existing methods in the literature. The computational results show that the proposed procedure for producing starting incumbents could act as a good heuristic for the RLP. Our B&C approaches significantly outperformed the available exact algorithm in the literature, especially for large scale instances with up to 500 nodes. Over the 400 test instances, two versions of our B&C approach were able to find optimal solutions in 289 and 307 instances, whereas the available exact algorithm solved only 180 instances. Our approaches significantly extends the ability to solve large RLP instances. The available exact algorithm was able to find optimal solutions to 300-node instances with $p = 10$ percent, whereas our approaches could solve 500-node instances with $p = 10$ percent and $p = 30$ percent. Because of the equivalence between the RLP, MCDSP and MLSTP, we then compared our two approaches with nine other exact algorithms using 47 benchmark MCDSP/MLSTP instances. Our computational results show our B&C approaches are efficient exact algorithms for the MCDSP and MLSTP. Our both approaches found optimal solutions in 45 of the 47 instances. In comparison with other available algorithms, our algorithms attained a higher success rate with less running time. Our computational results over 447 benchmark instances demonstrate that the B&C approach benefits from the lifted cuts derived from the polyhedral results.

Acknowledgment

We would like to express our sincere appreciation to the editor and the referees in making valuable comments and suggestions to the paper. The authors thank Professors Chen, Ljubić, Raghavan, Gendron, Lucena, and Cunha for providing their test instances. The research of the first author has been partially supported by the Natural Science Foundation of China (grant nos. 71101105 and 71532015), the Ministry of Education Humanities and Social Science Project (grant no. 15YJA630030), Program for New Century Excellent Talents in University (grant no. NCET-13-0424), and the Fundamental Research Funds for the Central Universities. The research of the second author was supported by a discovery research grant from NSERC, the Natural Science Engineering Research council of Canada (grant no. 4138).

References

Agarwal, Y., & Aneja, Y. (2012). Fixed-charge transportation problem: Facets of the projection polyhedron. *Operations Research, 60*(3), 638–654.
Álvarez-Miranda, E., Ljubić, I., & Mutzel, P. (2013). *Facets of combinatorial optimization: Festschrift for Martin Grötschel* (pp. 245–270). Berlin, Heidelberg: Springer.
Aneja, Y. (2012). Regenerator placement problem. In *Presentation on the 54th annual conference of the Canadian Operational Research Society (CORS2012), June 11–13, 2012*. Niagara Falls, Ontario.
Balas, E., & Ng, S. (1989). On the set covering polytope: I. All the facets with coefficients in {0, 1, 2}. *Mathematical Programming, 43*(1–3), 57–69.
Buchanan, A., Sung, J., Boginski, V., & Butenko, S. (2014). On connected dominating sets of restricted diameter. *European Journal of Operational Research, 236*(2), 410–418.
Chandran, B., & Hochbaum, D. (2009). A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations Research, 57*(2), 358–376.
Chen, S., Ljubić, I., & Raghavan, S. (2010). The regenerator location problem. *Networks, 55*(3), 205–220.
Chen, S., Ljubić, I., & Raghavan, S. (2012). About test instances of the RLP. Personal communication.
Chen, S., Ljubić, I., & Raghavan, S. (2015). The generalized regenerator location problem. *INFORMS Journal on Computing, 27*, 204–220.
Duarte, A., Martí, R., Resende, M., & Silva, R. (2014). Improved heuristics for the regenerator location problem. *International Transactions in Operational Research, 21*(4), 541–558.
Fan, N., & Watson, J. (2012). Solving the connected dominating set problem and power dominating set problem by integer programming. In G. Lin (Ed.), *Combinatorial optimization and applications*. In *Lecture notes in computer science: vol. 7402* (pp. 371–383). Berlin, Heidelberg: Springer.
Fernau, H., Kneis, J., Kratsch, D., Langer, A., Liedloff, M., Raible, D., & Rossmanith, P. (2011). An exact algorithm for the maximum leaf spanning tree problem. *Theoretical Computer Science, 412*(45), 6290–6302.
Fischetti, M., Leitner, M., Ljubić, I., Luipersbeck, M., Monaci, M., Resch, M., Salvagnin, D., & Sinnl, M. (2014). Thinning out steiner trees: A node-based model for uniform edge costs. In *11th DIMACS implementation challenge in collaboration with ICERM: Steiner tree problems, December 4–5, 2014*. Rhode Island, USA.
Flammini, M., Marchetti-Spaccamela, A., Monaco, G., Moscardelli, L., & Zaks, S. (2011). On the complexity of the regenerator placement problem in optical networks. *IEEE-ACM Transactions on Networks, 19*(2), 498–511.
Ford, L. R., & Fulkerson, D. R. (1974). *Flows in networks* (6th). Princeton University Press.
Fügenschuh, A., & Fügenschuh, M. (2008). Integer linear programming models for topology optimization in sheet metal design. *Mathematical Methods of Operations Research, 68*(2), 313–331.
Fujie, T. (2003). An exact algorithm for the maximum leaf spanning tree problem. *Computers & Operations Research, 30*(13), 1931–1944.
Fujie, T. (2004). The maximum-leaf spanning tree problem: Formulations and facets. *Networks, 43*(4), 212–223.
Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Boston, New York: Freeman.
Gendron, B., Lucena, A., Cunha, A., & Simonetti, L. (2014). Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing, 26*, 645–657.
Gendron, B., Scutellà, M., Garroppo, R., Nencioni, G., & Tavanti, L. (2016). A branch-and-benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research, 255*(1), 151–162.
Gouveia, L., Patrício, P., De Sousa, A., & Valadas, R. (2003). MPLS over WDM network design with packet level QoS constraints based on ILP models. In *INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications. IEEE societies: vol. 1* (pp. 576–586).
Guha, S., & Khuller, S. (1998). Approximation algorithms for connected dominating sets. *Algorithmica, 20*(4), 374–387.
Hartstein, I., Shalom, M., & Zaks, S. (2013). On the complexity of the regenerator location problem – treewidth and other parameters. In T. Erlebach, & G. Persiano (Eds.), *Approximation and online algorithms*. In *Lecture notes in computer science: vol. 7846* (pp. 42–55). Berlin, Heidelberg: Springer.
Hochbaum, D. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research, 56*(4), 992–1009.
Lucena, A., Maculan, N., & Simonetti, L. (2010). Reformulation and solution algorithms for the maximum leaf spanning tree problem. *Comput. Management Science, 7*(3), 289–311.
Mertzios, G., Sau, I., Shalom, M., & Zaks, S. (2012). Placing regenerators in optical networks to satisfy multiple sets of requests. *IEEE-ACM Transactions on Networks, 20*(6), 1870–1879.
Mertzios, G., Shalom, M., Wong, P., & Zaks, S. (2011). Online regenerator placement. In A. Fernndez Anta, G. Lipari, & M. Roy (Eds.), *Principles of distributed systems*. In *Lecture notes in computer science: vol. 7109* (pp. 4–17). Berlin, Heidelberg: Springer.
Mitchell, J. E. (2009). Integer programming: Branch and cut algorithms. In *Encyclopedia of optimization* (pp. 1643–1650). Berlin: Springer.
Pachnicke, S., Paschenda, T., & Krummrich, P. M. (2008). Physical impairment based regenerator placement and routing in translucent optical networks. In *Proc. 2008 optical fiber communication/national fiber optic engineers conference (OFC/NFOEC 2008), San Diego, CA, IEEE, Piscataway, NJ* (pp. 1–3).
Pedrola, O., Careglio, D., Klinkowski, M., Velasco, L., Bergman, K., & Solé–Pareta, J. (2013). Metaheuristic hybridizations for the regenerator placement and dimensioning problem in sub-wavelength switching optical networks. *European Journal of Operational Research, 224*(3), 614–624.
Rahman, Q. (2012). *Optimization of WDM optical networks*. Windsor, Ontario: University of Windsor (Ph.D. thesis). Unpublished doctoral dissertation.
Rahman, Q., Bandyopadhyay, S., & Aneja, Y. (2015). Optimal regenerator placement in translucent optical networks. *Optical Switching and Networking, 15*, 134–147.
Reingold, E., Nievergelt, J., & Deo, N. (1977). *Combinatorial algorithms: Theory and practice*. Prentice Hall College Div.
Sen, A., Murthy, S., & Bandyopadhyay, S. (2008). On sparse placement of regenerator nodes in translucent optical network. In *Proc. 2008 IEEE globecom conf., New Orleans, Louisiana, USA, IEEE, Piscataway, NJ* (pp. 1–6).
Shalom, M., Voloshin, A., Wong, P., Yung, F., & Zaks, S. (2012). Online optimization of busy time on parallel machines. In M. Agrawal, S. Cooper, & A. Li (Eds.), *Theory and applications of models of computation*. In *Lecture notes in computer science: vol.7287* (pp. 448–460). Berlin, Heidelberg: Springer.
Simmons, J. (2006). Network design in realistic “all-optical” backbone networks. *IEEE Communications Magazine, 44*(11), 88–94.

Simonetti, L., Salles da Cunha, A., & Lucena, A. (2011). The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In J. Pahl, T. Reiners, & S. Voß (Eds.), *Network optimization*. In *Lecture notes in computer science: vol. 6701* (pp. 162–169). Berlin, Heidelberg: Springer.

Sriram, K., Griffith, D., Su, R., & Golmie, N. (2004). Static vs. dynamic regenerator assignment in optical switches: models and cost trade-offs. In *Proc. 2004 workshop on high performance switching and routing, Phoenix, Arizona, IEEE, Piscataway, NJ* (pp. 151–155).

Wolsey, L. A. (1998). *Integer programming*. New York: John Wiley and Sons, Inc.

Yang, X., & Ramamurthy, B. (2005a). Dynamic routing in translucent wdm optical networks: The intradomain case. *Journal of Lightwave Technology, 23*(3), 955–971.

Yang, X., & Ramamurthy, B. (2005b). Sparse regeneration in translucent wavelength-routed optical networks: Architecture, network design and wavelength routing. *Photonic Network Communications, 10*(1), 39–53.

Yetginer, E., & Karasan, E. (2003). Regenerator placement and traffic engineering with restoration in GMPLS networks. *Photonic Network Communications, 6*, 139–149.

Yue, C., Li, X., Wei, K., & Lin, S. (2014). Heuristics for the regenerator location problem. In *2014 international conference on computer, network security and communication engineering: vol. 7* (pp. 641–647). Shenzhen, China: DEStech Publications, Inc.