INFORMS Journal on Computing
Vol. 27, No. 2, Spring 2015, pp. 204–220
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2014.0621
© 2015 INFORMS

The Generalized Regenerator Location Problem

Si Chen
Arthur J. Bauernfeind College of Business, Murray State University, Murray, Kentucky 42071,
si.chen@murraystate.edu

Ivana Ljubić
Department of Statistics and Operations Research, University of Vienna, Vienna, 1090 Austria,
ivana.ljubic@univie.ac.at

S. Raghavan
Smith School of Business and Institute for Systems Research, University of Maryland, College Park, Maryland 20742,
raghavan@umd.edu

In an optical network a signal can only travel a maximum distance $d_{\max}$ before its quality deteriorates to the point that it must be regenerated by installing regenerators at nodes of the network. As the cost of a regenerator is high, we wish to deploy as few regenerators as possible in the network, while ensuring all nodes can communicate with each other. In this paper we introduce the *generalized regenerator location problem* (GRLP) in which we are given a set S of nodes that corresponds to candidate locations for regenerators, and a set T of nodes that must communicate with each other. If $S = T = N$, we obtain the regenerator location problem (RLP), which we have studied previously and shown to be NP-complete. Our solution procedure to the RLP is based on its equivalence to the maximum leaf spanning tree problem (MLSTP). Unfortunately, this equivalence does not apply to the GRLP, nor do the procedures developed previously for the RLP. To solve the GRLP, we propose reduction procedures, two construction heuristics, and a local search procedure that we collectively refer to as a heuristic framework. We also establish a correspondence between the (node-weighted) directed Steiner forest problem and the GRLP. Using this fact, we provide several ways to derive natural and extended integer programming (IP) and mixed-integer programming (MIP) models for the GRLP and compare the strength of these models. Using the strongest model derived on the natural node selection variables we develop a branch-and-cut approach to solve the problem to optimality. The results indicate that the exact approach can easily solve instances with up to 200 nodes to optimality, whereas the heuristic framework is a high-quality approach for solving large-scale instances.

Keywords: optical network design; extended formulations; multicommodity flow; branch and cut; heuristics regenerator location
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received August 2012; revised December 2013, June 2014; accepted July 2014.

1. Introduction

In today's society the Internet has become ubiquitous. The number of services and applications based on Internet protocol (IP) have grown exponentially over the last decade. Much of this growth has been fueled by modern optical networks that provide high capacity transport infrastructure for advanced digital services. This paper deals with an important and fundamental problem concerning the geographical reach of an optical network. An optical signal can only travel a maximum distance (say $d_{\max}$) before its quality deteriorates (due to transmission impairments in the fiber) and needs to be regenerated. To accomplish this, *regenerators* that convert the optical signal to an electronic one (using 3R regeneration to reamplify, reshape, and retime the signal, see Borella et al. 1997, Mukherjee 2000, Zymolka 2006) and then back to an optical one are installed at nodes of the optical network. Traditionally, optical networks have been designed to be *opaque*, in the sense that regenerators have been installed at every node of the network. However, this strategy is expensive and energy consuming (because each optical signal is converted to an electrical one and back to an optical one at each node of the network). To lower the costs of the optical network, since the cost of regenerators is very high (for example, Mertzios et al. 2012, indicate a cost of $160,000 per regenerator), we wish to instead design a translucent optical network (Rumley and Gaumier 2009) and deploy regenerators at a subset of nodes in the network (i.e., to deploy fewer regenerators) while ensuring all nodes can communicate with each other (i.e., send a signal to each other).

In this context our paper studies the following network design problem that we refer to as the *generalized regenerator location problem* (GRLP). We are given a network $G = (N, F)$ where N denotes the set of nodes and F denotes the set of edges. Set $S \subseteq N$ is the candidate locations where regenerators can be placed, $T \subseteq N$ is the set of terminal nodes that must communicate with each other. A mapping $\ell\colon F \to \mathbb{R}^+$ defines the *length* of edges. A maximum distance of $d_{\max} > 0$ determines how far a signal can traverse before its quality deteriorates and needs to be regenerated. Our goal is to determine a minimum cardinality subset of nodes $L \subseteq S$ such that for every pair of nodes in T there exists a simple path in G with the property that there is no subpath (i.e., a subsequence of edges on the path) with *length* $> d_{\max}$ without regenerators placed on its internal nodes. (The length of a simple path P between u and v is defined as the sum of lengths of its edges, and the nodes other than u and v visited along that path are called *internal nodes*.)

When $S = T = N$, we obtain a special case of the GRLP that we refer to as the *regenerator location problem* (RLP). In the RLP, all nodes serve as terminals as well as candidate locations for placement of regenerators. In our earlier work (see Chen et al. 2010), we introduced the RLP and presented three heuristics and a branch-and-cut approach for it. In particular, we established a correspondence between the RLP and the *maximum leaf spanning tree problem* (MLSTP). In this paper, we focus on the GRLP (which turns out to be significantly more challenging) for several reasons. In practice it is not necessarily the case that all nodes in the network need to communicate with each other (i.e., $T \neq N$). Furthermore, for administrative reasons (e.g., ease of maintenance) a service provider may wish to restrict the set of locations where regenerators may be installed (i.e., $S \neq N$). Thus, the solution to the RLP may not accurately reflect some of the practical constraints that a service provider may wish to ensure (in the first situation the solution to the RLP may install more regenerators than necessary to ensure all nodes in N can communicate, whereas in the second situation it may install a regenerator at a node where it may not be desirable to do so). Unfortunately, for the GRLP, the correspondence with the MLSTP does not hold. Consequently, our earlier work for the RLP cannot be applied to the GRLP.

Although regenerator placement is a significant issue in optical network design, prior to our earlier work in Chen and Raghavan (2007), the RLP seems to have been relatively ignored by the academic literature on telecommunications network design. At the time of submission of our earlier work (Chen et al. 2010) we had only come across two papers (Gouveia et al. 2003, Yetginer and Karasan 2003) that discussed the issue of regenerator placement within the context of a larger network design problem. Since then, interest in the RLP has grown significantly and additional papers have appeared in the literature that discuss the RLP (Bathula et al. 2013, Flammini et al. 2011, Lucerna et al. 2009, Mertzios et al. 2012, Pachnicke et al. 2008), although still in the restrictive form where regenerators can be placed at every node of the network.

Our Contribution. In this paper, we develop a mathematical programming approach for the GRLP by establishing a connection between the GRLP and a particular node-weighted directed Steiner forest problem (NWDSFP). With this connection, we devise a simple cut formulation in the space of the natural node selection variables. We then describe two equivalent extended formulations (a typical multicommodity and directed cut formulation) for the GRLP. Using a technique of disaggregation we develop a stronger extended formulation for the GRLP. We show this disaggregation is equivalent to a node-splitting approach, and show that this node-splitting model provides a stronger model for the GRLP using the natural variables—specifically a model that contains separating node-cuts in the space of the natural node selection variables. Because the separating node-cut model provides a strong formulation with the fewest number of variables (which can make quite a significant difference in terms of computational performance), we develop a branch-and-cut (B&C) approach on this model to provide exact solutions for the GRLP. This approach is quite amenable to solving to optimality instances with up to 200 nodes. We also devise a preprocessing procedure, two construction heuristics, and a local search procedure. Two heuristic frameworks are proposed, in which the preprocessing, followed by one of the two construction heuristics and the local search procedure, is applied. These heuristic frameworks are extremely fast and produce high-quality solutions for the GRLP. Specifically, of 450 instances studied in this paper (ranging from 50 to 500 nodes) the B&C approach solves 346 instances to optimality within the given time limit of one hour per instance. Of these 346 instances our heuristic frameworks found the optimal solution 307 times. Even for the largest 500-node problems, our heuristic frameworks typically run in about half a minute per instance.

Organization of the Paper. The rest of this paper is organized as follows. Before we conclude this introductory section we provide a brief literature review on the RLP. In §2 we describe a graph transformation procedure that greatly simplifies conceptualization of the GRLP, and propose a set of preprocessing steps to reduce the size of GRLP instances. Section 3 discusses both natural and extended formulations for the GRLP, shows how to strengthen them, and establishes the equivalence of the different formulations. Section 4 describes two heuristics and a local search procedure for the GRLP. Section 5 describes the weighted GRLP and explains how to adapt our B&C procedure and heuristics to it. Section 6 presents our computational experiments, and §7 provides concluding remarks.

1.1. Related Literature

Regenerator technology and its use within an optical network has been well known for quite a long time. However, until our earlier work (Chen et al. 2010), the RLP does not seem to have been considered as a stand alone network design problem within the academic research community. Frequently, in optical network design problems, constraints related to regeneration are ignored and dealt with once routing paths have been determined for optical transmission, or costs associated with regenerators were ignored (Gouveia et al. 2003). Because network design is often done in a hierarchical fashion, addressing the RLP or GRLP at the outset of the network design planning process ensures that regenerators are placed at nodes of the network so that all nodes of the network that need to communicate may communicate without worry of physical impairments of the signal. This greatly simplifies the design process, and is extremely useful for telecommunications managers.

In Chen et al. (2010) we established a correspondence between the RLP and MLSTP. We then devised three heuristics and a branch-and-cut algorithm for the RLP. The branch-and-cut approach was based on formulating the RLP as a Steiner arborescence problem (SAP) with a unit degree on the root node on a directed graph in which nodes are split into arcs. Due to the correspondence between the RLP and MLSTP, these procedures are equally applicable to the MLSTP as well. Independently and subsequent to our work, Lucena et al. (2010) considered the MLSTP and proposed two different formulations for it. The first one is based on a model developed by Fernandes and Gouveia (1998) for the minimum spanning tree problem with a constraint on the number of leaves. The second one is identical to the SAP model proposed in Chen et al. (2010). Lucena et al. (2010) also (independently) proposed a heuristic for the MLSTP, which is identical to the heuristic called H1 in Chen et al. (2010).

Flammini et al. (2011) developed an approximation algorithm for the RLP with approximation ratio $O(\log n)$. They showed that the RLP is not approximable in polynomial time with an approximation factor better than $(1-\epsilon)\log n$. Sen et al. (2010) point out some errors in Flammini et al. (2011). By making the connection between the RLP and the minimum connected dominating set problem (MCDSP)[1] the authors show that there is a $(\log\delta+2)$-approximation algorithm for the RLP, where δ denotes the maximum degree of the input graph. In some design scenarios, rather than focusing on minimizing the number of regenerator locations, the focus is on minimizing the total number of regenerators installed with the provision that at each location, multiple regenerators need to be installed (one for each pair of nodes that communicates through that location and needs a regenerator at that location). Mertzios et al. (2012) studied the complexity of this problem and showed that it does not admit a polynomial-time approximation scheme. While the papers discussed so far focus solely on the location of regenerators, recently, Patel et al. (2010) considered the regenerator placement and traffic grooming problem together. In other words, in addition to determining the location of regenerators, they also determine a logical topology of the network (links on this logical topology are lightpaths) and route traffic over this logical topology. They described separate heuristics for the RLP and traffic grooming problem, and discussed how to combine the two heuristics to consider both problems together.

Several recent papers discuss extensions of the RLP. Mertzios et al. (2011) introduce an online version of the RLP. Bathula et al. (2013) consider the RLP with additional routing restrictions. Yildiz and Karasan (2015) add node-survivability restrictions to the RLP. In one variant the solution has to remain connected after failure of any single regenerator node, and in another variant, the solution has to remain connected after failure of any single node in the network. To the best of our knowledge, the GRLP has not been considered previously in the literature.

2. Preliminaries

In Chen et al. (2010) we showed that the RLP is NP-complete. Since the GRLP generalizes the RLP, its NP-completeness follows immediately. Alternatively, to show the NP-completeness of the GRLP directly, one might consider a transformation from the hitting set problem as shown in Chen et al. (2009).

Before solving the GRLP, it is useful to consider a graph transformation into a *communication graph B* that is described in this section. We also address questions related to checking the feasibility of the input graph. Finally, in this section we show how to efficiently reduce the size of the input graph, by applying several reduction procedures.

2.1. The Communication Graph

Given a graph $G=(N,F)$ with edge lengths ℓ, let $d(u,v)$ denote the length of the shortest path between nodes u and v in G. We now generate a graph $B=(N,E)$, called the *communication graph*, such that $E=\{e=\{u,v\}\mid u,\ v\in N,\ d(u,v)\le d_{\max}\}$. In other words, if

[1] This connection is also implicit from the correspondence between the RLP and MLSTP shown in Chen et al. (2010) and the correspondence between the MLSTP and MCDSP shown in Lucena et al. (2010).

the length of the shortest path $d(u, v)$ between a pair of nodes (u, v) in G is $\leq d_{\max}$, we construct edge $\{u, v\}$ in B. Observe that if every pair of nodes in T has an edge connecting them in B, then no regenerators are required. On the other hand, every node pair in T that is not connected by an edge in B requires regenerators to communicate. We call such node pairs *not directly connected* or *NDC node pairs*. Without loss of generality, we will assume that the NDC node pairs are defined as (t_1, t_2), where $t_1, t_2 \in T$ and $t_1 < t_2$. The set of all NDC node pairs will be denoted by NDC. It should be clear that it suffices to consider the GRLP on the communication graph B. In other words, given the communication graph B, the GRLP on B searches for the minimum cardinality set of nodes $L \subseteq S$ to place regenerators, such that for every NDC node pair in T there exists a path with regenerators placed at all its internal nodes.

Lemma 1. *Without loss of generality, we can assume that for any GRLP instance, $\{S, T\}$ is a proper partition of N, i.e., $S \cap T = \varnothing$, $S \cup T = N$, and $S, T \neq \varnothing$.*

Proof. Assume that in a graph G we have that $S \cup T \neq N$. The nodes from $N \backslash \{S \cup T\}$ will not be candidates for placing regenerators, nor will they be part of a NDC node pair. Since the communication graph B already specifies nodes that can communicate without regenerators, the nodes in $N \backslash \{S \cup T\}$ can be deleted from B.

Assume now that $S \cap T \neq \varnothing$. We show how to transform B into a communication graph $B' = (N', E')$ in which the set of potential regenerator locations (S') and the set of terminal nodes (T') are disjoint, without changing the value of the solution. To do so, we basically split every node $i \in T \cap S$ into a node $s_i \in S'$ and a node $t_i \in T'$ and reconnect them as follows:

$$\begin{aligned}
N' &= \{s_i \mid i \in T \cap S\} \cup \{t_i \mid i \in T \cap S\} \cup \{j \mid j \notin T \cap S\}, \\
E' &= \big\{\{s_i, s_j\}, \{t_i, t_j\}, \{s_i, t_j\}, \{s_j, t_i\} \mid \{i, j\} \in E,\, i, j \in T \cap S\big\} \\
&\quad \cup \big\{\{s_i, j\}, \{t_i, j\} \mid \{i, j\} \in E,\, i \in T \cap S,\, j \notin T \cap S\big\} \\
&\quad \cup \big\{\{i, j\} \mid \{i, j\} \in E,\, i, j \notin T \cap S\big\}.
\end{aligned}$$

Figures 1(a) and 1(b) illustrate this transformation. It is easy to verify that any feasible solution on B can be transformed into an equivalent one on B' with the same number of regenerators, and vice versa. □

Given the previous lemma, in the rest of the paper we will assume that the GRLP is posed on a communication graph, and the sets S and T form a proper partition of N. We call nodes in S, *s-nodes*, and nodes in T, *t-nodes*. Any maximally connected component consisting of s-nodes only will be called an *S-component*. The number of all S-components in B will be denoted by n_S. Given a node $u \in T$ and an arbitrary set $\hat{N} \subseteq N$, we define $\hat{N}$-degree of u as $\deg_{\hat{N}}(u) = |\{\{u, v\} \mid v \in \hat{N}, v \neq u\}|$. If $\deg_{\hat{N}}(u) = 1$, we say that u has a *unit $\hat{N}$-degree*.

2.2. Checking Feasibility of the GRLP

Observe that the necessary and sufficient condition for an instance of the GRLP to be feasible is that the two end points of every NDC node pair are connected to at least one common S-component. For the input graph B' given in Figure 1(b), we identify two S-components of B' and replace them with *super-nodes* S_1 and S_2, as shown in Figure 1(c). This instance is feasible since the two end points of every NDC node pair are connected to at least one common S-component.

To check feasibility of a GRLP instance, one has to consider all NDC pairs (there are $O(|T|^2)$ of them in the worst case) and check that they are connected to a common S-component ($n_S \in O(|S|)$). Therefore, the feasibility check can be done in $O(|T|^2 \cdot |S|)$ time. For the rest of the paper, we will assume that a given instance of the GRLP is feasible.

2.3. Preprocessing for the GRLP

Observe that in the communication graph B the following properties hold.

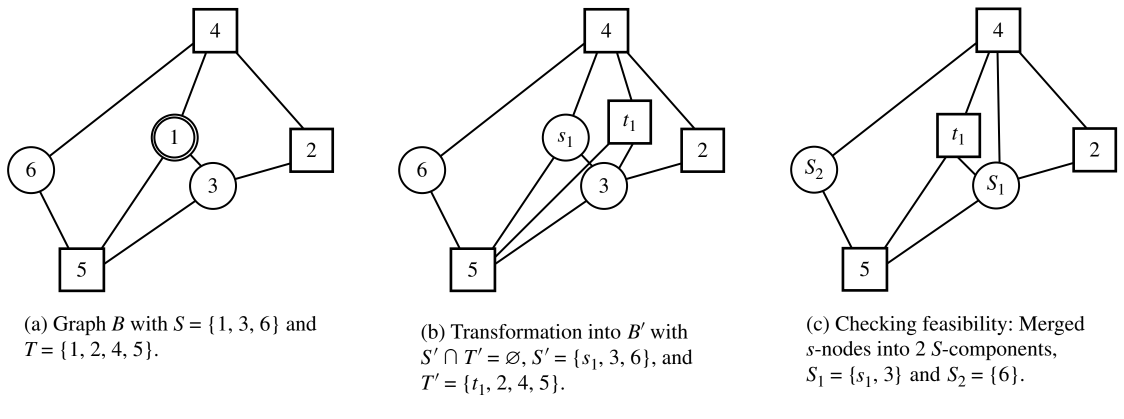

(a) Graph B with $S = \{1, 3, 6\}$ and $T = \{1, 2, 4, 5\}$.
(b) Transformation into B' with $S' \cap T' = \varnothing$, $S' = \{s_1, 3, 6\}$, and $T' = \{t_1, 2, 4, 5\}$.
(c) Checking feasibility: Merged s-nodes into 2 S-components, $S_1 = \{s_1, 3\}$ and $S_2 = \{6\}$.

Figure 1 Eliminating Nodes from $S \cap T$ in the Communication Graph B and Checking Feasibility

(P0) If an s-node i has unit degree, removal of node i from B will not change the problem;

(P1) If all the neighbors of an s-node i are connected to each other, removal of node i from B will not change the problem;

(P2) If a t-node i is not in any NDC node pair, removal of node i from B will not change the problem;

(P3) If a t-node i has unit S-degree (i.e., it is only connected to one s-node l), every feasible solution must include a regenerator deployed at the s-node connected to node i (i.e., node l);

(P4) If the two end points of a NDC node pair (i, k) are connected to only one common S-component, say $S_j \subseteq S$, and i (or k) has unit S_j-degree, any feasible solution must deploy a regenerator at the corresponding adjacent node $l \in S_j$.

We now propose a preprocessing procedure that repeatedly uses the properties (P0)–(P4) to reduce the input graph B. Let us denote by $L \subseteq S$, the set of locations where regenerators are placed, and by $Q \subseteq \{1, \ldots, n_S\}$ the set of indices of S-components in which regenerators are placed. When removing a node u from B, we will denote it by $B \backslash u$. In that case, the node will be removed from N (and, correspondingly, from T or S), and all its neighboring edges will be removed from E. The psuedo-code of the preprocessing procedure is given in the online supplement (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2014.0621).

In each preprocessing iteration, we check the properties (P0)–(P4), and, if the set L of regenerators that can be fixed is nonempty, our preprocessor uses a subroutine called UPDATE(B, L, Q) whose pseudocode is provided in the online supplement. The procedure essentially identifies all the node pairs that can be connected after deploying regenerators in L, and adds to B the edges associated with these node pairs. Thereby, the connectivity of B is iteratively improved and the number of NDC node pairs reduced. In addition, the UPDATE procedure deletes nodes from B where we have fixed the location of regenerators. This can significantly reduce the number of s-nodes of the communication graph B. The UPDATE procedure is used later in §4 within the construction heuristics and the post-optimization.

Lemma 2. *The running time of the preprocessing procedure is $O(|S|^2|N|^2)$.*

Proof. See online supplement. □

3. Mathematical Programming Approach

In this section we first show how to transform a GRLP instance given on B into a *special instance* of the node-weighted directed Steiner forest problem (NWDSFP) on an auxiliary graph H. Using this transformation, we study a natural formulation (in the space of node selection variables) and two extended formulations for the GRLP. We show how to strengthen one of the extended formulations by a disaggregation technique. We also obtain this strengthening in the natural formulation (and the other extended flow formulation) by "splitting" s-nodes. Using the strengthened formulation on the natural variables, we propose a branch-and-cut algorithm that is described in §3.4.

3.1. Transforming the GRLP Into a Particular Node-Weighted Directed Steiner Forest Problem

When modeling the GRLP, we want to find a solution such that any NDC node pair (t_1, t_2) is connected via a path where all the internal nodes are from S. To forbid t-nodes along a path between any two NDC node pairs, we will work on a directed graph H in which t-nodes will have either in- or out-degree equal to zero. The construction of H is described in the following. During this transformation, we create a set of *origin-destination pairs*, denoted by D that correspond to the NDC node pairs from B. The directed graph $H = (V_H, A_H)$ is defined as follows:

$$V_H = V_O \cup V_D \cup S,$$

where V_O and V_D are created as follows.

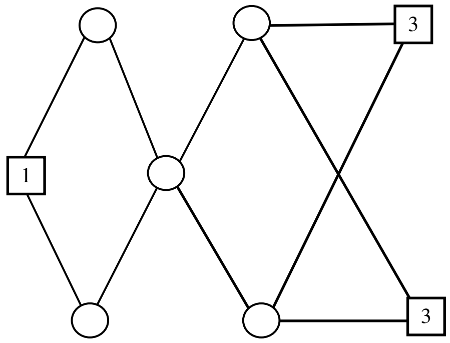
For $v \in T$, such that there exists a $(v, l) \in \text{NDC}$, create a node $v_0 \in V_0$. For each $v \in T$, such that there exists a $(l, v) \in \text{NDC}$, create a node $v_d \in V_D$. The arc set

$$A_H = \big\{(i, j) \mid \{i, j\} \in E,\ i \in V_O, j \in S,\\ \text{or } i \in S,\ j \in V_D,\ \text{ or } i, j \in S\big\}.$$

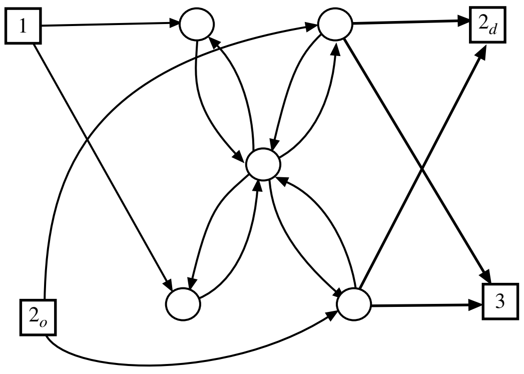
The cost of each arc $(i, j) \in A_H$ is defined as $c_{ij} = 0$. Node weights are defined as: $w_i = 1$ if $i \in S$, and $w_i = 0$ if $i \in V_D \cup V_O$. Figure 2 illustrates this transformation. We now define the NWDSFP as follows. Given a directed graph $H = (V_H, A_H)$, a collection of origin-destination pairs $D \subseteq V_H \times V_H$, and node and arc weights, w: $V_H \to \mathbb{R}^+$ and c: $A_H \to \mathbb{R}^+$, respectively, the *node-weighted directed Steiner forest problem* (NWDSFP) searches for a subgraph $H' = (V', A')$ of H that contains a directed path for every pair of nodes from D and that minimizes $\sum_{i \in V'} w_i + \sum_{a \in A'} c_a$.

One can easily show that the GRLP can be transformed into a special instance of the NWDSFP (one where arc costs are zero and the set of origins and destinations are disjoint) on the graph H obtained as described earlier.

Lemma 3. *Given a feasible solution to a GRLP instance on the communication graph B it can be transformed to a feasible solution of equal cost to the instance of the NWDSFP on the graph H, and vice versa.*



(a) Communication graph B with NDC node pairs (1, 2), (1, 3), and (2, 3).



(b) Directed graph H with $V_O = \{1, 2_o\}$, $V_D = \{2_d, 3\}$, and the set of NDC node pairs $D = \{(1, 2_d), (1, 3), (2_o, 3)\}$.

Figure 2 Transformation of a GRLP Instance into a NWDSFP Instance

3.2. Natural and Extended Formulations on H

For each origin node $r \in V_O$, denote by $V_D(r) = \{i \in V_D \mid (r, i) \in D\}$ the set of its destination nodes. A solution to the GRLP on H can be seen as a union of Steiner arborescences rooted at r with $V_D(r)$ nodes as terminals (that are leaves).

We can introduce a natural formulation on the node selection variables, as follows. We create binary variables $y_i \in \{0, 1\}, i \in S$ that are set to one if node $i \in S$ is in the solution (and 0 otherwise). Furthermore, we fix $y_i = 1$ for $i \in V_O \cup V_D$. (In this and in the following models, y variables assigned to $V_O \cup V_D$ can easily be removed from the formulation, but they are kept for the sake of better readability.) Given $W \subseteq V_H$, denote by $\delta^-(W) = \{(i, j) \mid i \notin W, j \in W\}$. The natural formulation is based on the observation that given a choice of binary values for the y_i variables, a feasible solution to the GRLP exists if after setting arc capacities to $\min\{y_i, y_j\}$ for every $(i, j) \in A_H$ the directed graph contains Steiner arborescences (from each origin node $r \in V_O$ to $V_D(r)$). The natural formulation then reads as:

$$(\mathrm{CUT}_y) \quad \min \sum_{i \in S} y_i \tag{1}$$

$$\text{s.t.} \sum_{(i,j) \in \delta^-(W)} \min\{y_i, y_j\} \geq 1 \quad \forall r \in V_O,$$
$$\forall W \subseteq V_H \backslash \{r\},\ W \cap V_D(r) \neq \varnothing, \tag{2}$$

$$y_i = 1 \quad \forall i \in V_O \cup V_D, \tag{3}$$

$$y_i \in \{0, 1\} \quad \forall i \in S. \tag{4}$$

Constraints (2) represent a compact way of writing $2^{|\delta^-(W)|}$ many inequalities per each constraint of type (2). Indeed, for any index set of arcs $I = \{1, \ldots, |I|\}$, inequality $\sum_{(i,j) \in I} \min\{y_i, y_j\} \geq 1$ can be replaced by the following collection of $2^{|I|}$ inequalities: $\sum_{(i,j) \in I, k = i \oplus j} y_k \geq 1$. (For example if $I = \{(1, 2), (3, 4)\}$, then the collection of inequalities is $y_1 + y_3 \geq 1$, $y_1 + y_4 \geq 1$, $y_2 + y_3 \geq 1$, and $y_2 + y_4 \geq 1$.)

Alternatively, one may also consider an extended formulation of CUT_y, by introducing binary arc variables $x_{ij} \in \{0, 1\}$, which will be set to one if the arc $(i, j) \in A_H$ is used along a path between an origin-destination pair $(o, d) \in D$, and to zero otherwise. Furthermore, for ease of notation, given any vector $\mathbf{x}$ over a ground set, and a subset of elements U in the ground set, let $x(U) = \sum_{(i,j) \in U} x_{ij}$ denote the sum of the elements of the vector taken over the subset U. In the corresponding model, which we will refer to as $\mathrm{CUT}_{x,y}$, inequalities (2) are replaced by:

$$x(\delta^-(W)) \geq 1 \quad \forall r \in V_O,$$
$$\forall W \subseteq V_H \backslash \{r\}, W \cap V_D(r) \neq \varnothing, \tag{5}$$

$$x_{ij} \leq \min\{y_i, y_j\} \quad \forall (i, j) \in A_H, \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A_H. \tag{7}$$

We now describe a compact extended multicommodity flow formulation on H. In this model, for every origin-destination pair $(o, d) \in D$, we create a commodity with unit requirement. Let f_{ij}^k represent the amount of flow of commodity k sent on arc $(i, j) \in A_H$. Notice that given a choice of binary values for the y_i variables, a feasible solution to the problem exists if after setting arc capacities to $\min\{y_i, y_j\}$ for every $(i, j) \in A_H$ the directed graph supports the flow. In the multicommodity flow model (MCF_y) inequalities (2) in CUT_y are replaced by:

$$\sum_{(i,j) \in A_H} f_{ij}^k - \sum_{(j,i) \in A_H} f_{ji}^k$$
$$= \begin{cases} -1, & i = d; \\ 1, & i = o \quad \forall i \in V_H, \forall k = (o, d) \in D; \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

$$0 \leq f_{ij}^k \leq \min\{y_i, y_j\} \quad \forall (i, j) \in A_H \, \forall k \in D. \tag{9}$$

Let $v_{\mathrm{LP}}(M)$ denote the value of the LP-relaxation of a MIP model M, P_M denote the feasible space of its linear relaxation, and let $\mathrm{Proj}_y(P_M)$ denote its projection onto the space of $(y_i)_{i\in S}$ variables. The following lemma easily follows in a straightforward fashion from the max-flow min-cut theorem (Ford and Fulkerson 1956).

Lemma 4. *The formulations* CUT_y, MCF_y, *and* $\mathrm{CUT}_{x,y}$ *are equally strong and their projections onto the space of the y variables are identical; i.e.,* $v_{\mathrm{LP}}(\mathrm{CUT}_y) = v_{\mathrm{LP}}(\mathrm{MCF}_y) = v_{\mathrm{LP}}(\mathrm{CUT}_{x,y})$ *and* $P_{\mathrm{CUT}_y} = \mathrm{Proj}_y(P_{\mathrm{MCF}_y}) = \mathrm{Proj}_y(P_{\mathrm{CUT}_{x,y}})$.

3.3. Strengthening

To build a stronger model, we first consider $\mathrm{CUT}_{x,y}$ and disaggregate node and arc variables as follows. To each origin node $r \in V_O$, we assign the set of binary arc and node variables, x_{ij}^r and y_i^r, respectively, that are set to one if the arborescence rooted at r uses these arcs/nodes to connect to its destination nodes $V_D(r)$. The formulation then reads as follows:

$(\mathrm{DCUT}_{x,y})$

$$\min \sum_{i\in S} y_i \tag{10}$$

$$\text{s.t. } x^r(\delta^-(W)) \geq 1, \quad \forall r \in V_O, \forall W \subseteq V_H\backslash\{r\}, \quad W \cap V_D(r) \neq \varnothing, \tag{11}$$

$$x^r(\delta^-(i)) = \begin{cases} y_i^r & \forall i \in S, \\ 1 & \forall i \in V_D(r), \end{cases} \quad \forall r \in V_O, \tag{12}$$

$$y_i^r \leq y_i \quad \forall r \in V_O \forall i \in S, \tag{13}$$

$$y_i = 1 \quad \forall i \in V_O \cup V_D, \tag{14}$$

$$x_{ij}^r \in \{0,1\} \quad \forall (i,j) \in A_H, \ \forall r \in V_O, \tag{15}$$

$$y_i^r, y_i \in \{0,1\} \quad \forall i \in V_H, \ \forall r \in V_O. \tag{16}$$

In the above, $\delta^-(\{i\})$ is shortened to $\delta^-(i)$. Notice that for each arborescence rooted at node r we have enforced the indegree constraints that state the indegree of each node of an arborescence is equal to one. This strengthens the model.

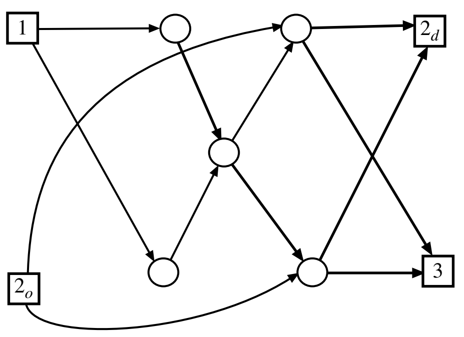
Lemma 5. *The model* $\mathrm{DCUT}_{x,y}$ *is strictly stronger than the model* $\mathrm{CUT}_{x,y}$. *In other words,* $v_{\mathrm{LP}}(\mathrm{DCUT}_{x,y}) \geq v_{\mathrm{LP}}(\mathrm{CUT}_{x,y})$ *and there are instances for which strict inequality holds.*

Proof. To prove this result it suffices to consider minimal feasible solutions $(\mathbf{x}'^r, \mathbf{y}'^r, \mathbf{y}') \in P_{\mathrm{DCUT}_{x,y}}$. Let $x'_{ij} = \max_r x'^r_{ij}$. Then it is easy to verify that $(\mathbf{x}', \mathbf{y}')$ is also feasible to inequalities (3) and (5). With regards to inequality (6), since $x'^r_{ij} \leq x'^r(\delta^-(j)) = y'^r_j$, $x_{ij} \leq y_j$ is automatically satisfied. In a minimal solution, for each arborescence at node r, $x'^r_{ij} \leq x'^r(\delta^-(i))$ (this is a consequence of max-flow min-cut). Since $x'^r_{ij} \leq x'^r(\delta^-(i)) = y'^r_i$, $x_{ij} \leq y_i$ is also automatically satisfied. Figure 3 illustrates an example that shows that $v_{\mathrm{LP}}(\mathrm{DCUT}_{x,y})$ can be strictly greater than $v_{\mathrm{LP}}(\mathrm{CUT}_{x,y})$. □

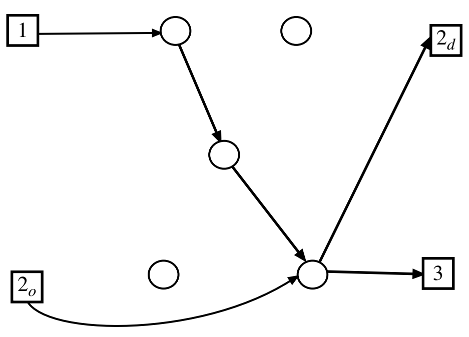
We now show how this same strengthening can be achieved using a node-splitting technique in the natural and extended flow formulation. The graph H is transformed into an *extended directed graph* $\tilde{H} = (\tilde{V}_H, \tilde{A}_H)$ such that:

$$\tilde{V}_H = V_O \cup V_D \cup S' \cup S'',$$
$$\text{where } S' = \{i' \mid i \in S\} \text{ and } S'' = \{i'' \mid i \in S\},$$
$$\tilde{A}_H = \tilde{A}_S \cup \tilde{A}, \quad \text{where } \tilde{A}_S = \{(i', i'') \mid i \in S\}, \text{ and}$$
$$\tilde{A} = \bigcup_{(i,j)\in A_H} \{(i'', j') \mid i, j \in S\} \cup \{(i'', j) \mid i \in S, j \in V_D\} \cup \{(i, j') \mid i \in V_O, j \in S\}.$$

In other words, for every node i in S we create two nodes i' and i'' corresponding to the node's input and output functions. Thereby, the cost of an arc $a \in \tilde{A}$ is set to zero, and the cost of an arc $a \in \tilde{A}_S$ is set to one.



(a) An optimal LP-solution of the model $\mathrm{CUT}_{x,y}$. $v_{\mathrm{LP}}(\mathrm{CUT}_{x,y}) = 2.5$. For all arcs shown above, $x_{ij} = 0.5$, and for all $i \in S$, $y_i = 0.5$. This solution is infeasible for $\mathrm{DCUT}_{x,y}$ because the s-node in the middle violates (13).



(b) An optimal LP-solution of the model $\mathrm{DCUT}_{x,y}$. $v_{\mathrm{LP}}(\mathrm{DCUT}_{x,y}) = 3$. For all arcs shown above, $x_{ij} = 1$, and for all nonisolated $i \in S$, $y_i = 1$.

Figure 3 Example Showing That $\mathrm{DCUT}_{x,y}$ Is Strictly Stronger Than $\mathrm{CUT}_{x,y}$

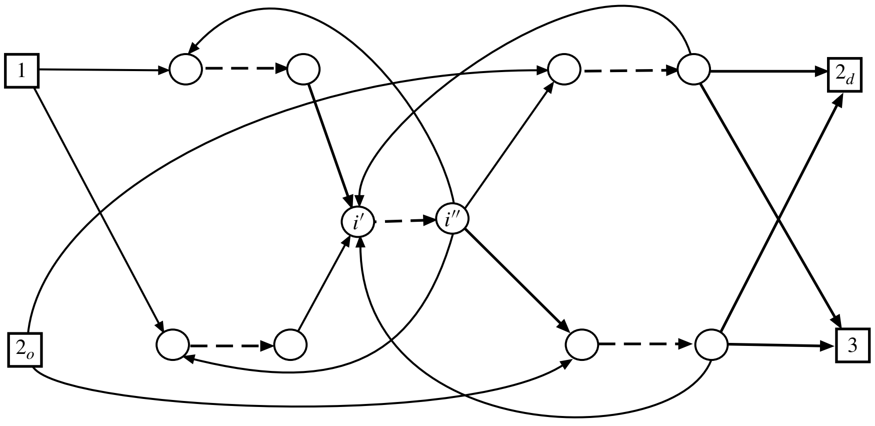


Figure 4 Node-Splitting Transformation of the Graph Given in Figure 2(b)
Note. Dashed arrows correspond to the s-nodes that are split.

Figure 4 illustrates this transformation and it is easy to see that the following result holds.

LEMMA 6. *Given a feasible solution to a GRLP instance on the communication graph B, it can be transformed to a feasible solution of equal cost to the directed Steiner forest problem (DSFP) on the extended directed graph $\tilde{H}$, and vice versa.*

Strengthened Node-Cut Formulation on H: For every arc $(i,j) \in \tilde{A}_H$, we introduce a binary variable $z_{ij} \in \{0,1\}$ that is set to 1 if the arc is used in establishing a connection between some $o \in V_O$ and $d \in V_D$ ($(o,d) \in D$), and 0 otherwise. The following cut-set formulation models multiple Steiner arborescences on $\tilde{H}$:

$$\min\left\{ \sum_{(i,j)\in\tilde{A}_S} c_{ij}z_{ij} \,\middle|\, \sum_{(i,j)\in\delta^-(W)} z_{ij} \geq 1,\ r \in V_O,\ W \subseteq \tilde{V}_H \setminus \{r\},\ W \cap V_D(r) \neq \varnothing, z_{ij} \in \{0,1\} \right\}.$$

We will refer to this model as SCUT. Later in this section, we will explain how SCUT yields the strengthened model in the natural space of the node variables.

Strengthened Flow Formulation on H: Alternatively, one could consider a multicommodity flow formulation on $\tilde{A}_H$, where commodities are defined identically to MCF_y. Instead of using the z_{ij} arc variables in the model we will use the node selection variables with the understanding that arc capacities are $\min\{y_i, y_j\}$ for the arcs in $\tilde{A}$ and y_i for the arcs in $\tilde{A}_S$. In other words, although we have conceptually split the nodes to obtain $\tilde{H}$, we will write the formulation on H. That yields the strengthened multicommodity flow formulation (MCF_y^+), which is identical to MCF_y with the additional inequality

$$\sum_{(j,i)\in A_H} f_{ji}^k \leq y_i \quad \forall\, i \in S,\ \forall\, k = (o,d) \in D. \tag{17}$$

We now show the equivalence of these alternate strengthened formulations for the GRLP.

THEOREM 1. *Formulations* SCUT, MCF_y^+, *and* $\text{DCUT}_{x,y}$ *are equally strong; i.e.,*

$$v_{\text{LP}}(\text{SCUT}) = v_{\text{LP}}(\text{MCF}_y^+) = v_{\text{LP}}(\text{DCUT}_{x,y}).$$

PROOF. First we show $v_{\text{LP}}(\text{DCUT}_{x,y}) \geq v_{\text{LP}}(\text{MCF}_y^+)$. Given any feasible solution $(\hat{\mathbf{x}}^r, \hat{\mathbf{y}}^r, \hat{\mathbf{y}}) \in \text{DCUT}_{x,y}$, we can construct a feasible solution to $(\hat{\mathbf{f}}, \hat{\mathbf{y}}) \in \text{MCF}_y^+$ as follows. The $\hat{\mathbf{y}}$ vector is set identically. All that remains is to argue that feasible flow values can be found that satisfy constraints (8), (9), and (17). Consider $\hat{x}_{ij}^r$ to represent the capacity of flow originating at node r on arc (i,j). Then, by max-flow min-cut arguments, it follows that it is possible to find values of flows ($\hat{\mathbf{f}}$) obeying the capacities such that an origin node $r \in V_O$ can send one unit of flow to each of its destination nodes in V_D. Consequently, constraint (8) is satisfied. Furthermore, for any commodity k, $\hat{f}^k(\delta^-(i)) \leq \hat{x}^r(\delta^-(i)) = \hat{y}_i^r \leq \hat{y}_i$ and constraint (17) is also satisfied. Noting that $\hat{f}_{ij}^k \leq \hat{f}^k(\delta^-(j))$ and constraint (17), we obtain $\hat{f}_{ij}^k \leq \hat{y}_j$. Also, $\hat{f}_{ij}^k \leq \hat{f}^k(\delta^+(i))$. But from constraint (8), $\hat{f}^k(\delta^+(i)) = \hat{f}^k(\delta^-(i))$; and from constraint (17), $\hat{f}^k(\delta^-(i)) \leq \hat{y}_i$. Thus, $\hat{f}_{ij}^k \leq \hat{y}_i$, and constraint (9) is satisfied. (This also indicates that in the presence of constraints (8) and (17), constraint set (9) is redundant and can be dropped from MCF_y^+.)

Next we show $v_{\text{LP}}(\text{MCF}_y^+) \geq v_{\text{LP}}(\text{SCUT})$. Given any feasible solution $(\hat{\mathbf{f}}, \hat{\mathbf{y}}) \in \text{MCF}_y^+$, we construct a solution $\hat{\mathbf{z}} \in P_{\text{SCUT}}$ by setting $\hat{z}_{i'i''} = \hat{y}_i$, for all $i \in S$, and $\hat{z}_{ij} = \max_k \hat{f}_{ij}^k$, otherwise. By max-flow min-cut arguments, it follows that on the extended directed graph $\tilde{H}$ with capacities equal to $\hat{\mathbf{z}}$, we are able to send one unit of flow from each $r \in V_O$ to each $d \in V_D(r)$. In other words, the solution $\hat{\mathbf{z}}$ is feasible for the model SCUT.

Finally, we show $v_{\text{LP}}(\text{SCUT}) \geq v_{\text{LP}}(\text{DCUT}_{x,y})$, which completes the proof. To prove this inequality, it suffices to consider minimal feasible solutions from P_{SCUT}. Given a minimal feasible $\hat{\mathbf{z}} \in P_{\text{SCUT}}$, we will show that there always exists a solution $(\hat{\mathbf{x}}^r, \hat{\mathbf{y}}^r, \hat{\mathbf{y}})$ feasible for the $\mathrm{DCUT}_{x,y}$ model, with the same objective value. We set $\hat{y}_i = \hat{z}_{i'i''}$, for all $i \in S$ and $\hat{y}_i = 1$, for $i \in V_D \cup V_O$. On the graph $\tilde{H}$ with arc capacities defined using $\hat{\mathbf{z}}$ values, we are able to send one unit of flow from every $r \in V_O$ to every $d \in V_D(r)$. Without explicitly stating the corresponding multicommodity flow formulation on $\tilde{H}$, let $f_{ij}^{(r,d)}$ denote the amount of flow for a commodity $(r,d) \in D$ sent along an arc $(i,j) \in \tilde{A}_H$. We now define the values of $\hat{\mathbf{x}}^r$ and $\hat{\mathbf{y}}^r$ as follows:

$$\hat{x}_{ij}^r = \begin{cases} \max_{d \in V_D(r)} f_{i''j'}^{(r,d)}, & i, j \in S; \\ \max_{d \in V_D(r)} f_{i,j'}^{(r,d)}, & i \in V_O,\ j \in S; \\ \max_{d \in V_D(r)} f_{i'',j}^{(r,d)}, & i \in S,\ j \in V_D; \end{cases}$$

$$r \in V_O,\ (i,j) \in A_H \quad \text{and} \quad \hat{y}_i^r = \hat{x}^r(\delta^-(i)),\ i \in S.$$

By construction of the flow and the definition of $\hat{\mathbf{x}}^r$, we also have that $\hat{x}^r(\delta^-(i)) = 1$, for all $r \in V_O$ and $i \in V_D(r)$ and hence, together with the definition of $\hat{\mathbf{y}}^r$, constraints (11) and (12) are satisfied. The constructed vector $(\hat{\mathbf{x}}^r, \hat{\mathbf{y}}^r, \hat{\mathbf{y}})$ also satisfies constraints (13), due to the minimality of the capacity vector $\hat{\mathbf{z}}$. Indeed, assume that our choice of the flow f causes a violation of the constraint $\hat{y}_i^r \le \hat{y}_i$ for some $r \in V_O$ and $i \in S$. This implies that, without loss of generality, there exist exactly two commodities, say $d_1, d_2 \in V_D(r)$, such that $f^{(r,d_1)}$ and $f^{(r,d_2)}$ are routed over (i', i'') in $\tilde{H}$, but they are not routed over the same collection of paths between r and i'. In that case, rerouting the flow of one of the two commodities still induces a feasible solution, with at least one $\hat{\mathbf{z}}$ capacity being reduced. But, this is in contradiction to our assumption that $\hat{\mathbf{z}}$ is a minimal feasible solution in P_{SCUT}. □

Given the specific cost structure of the GRLP, we are not interested in all possible cut sets separating the set of terminals $V_D(r)$ from its root $r \in V_O$, but rather on those cut sets among them consisting *solely of split arcs represented by* $\tilde{A}_S$. This means that for modeling the GRLP on the extended directed graph $\tilde{H}$, it is sufficient to change the objective function to $\sum_{(i',i'') \in \tilde{A}_S} z_{i'i''}$ and look at the following family of cut sets:

$$\mathcal{W} = \bigcup_{r \in V_O} \{ W \mid W \subseteq \tilde{V}_H \setminus \{r\},\ W \cap V_D(r) \neq \varnothing \text{ and } \delta^-(W) \cap \tilde{A} = \varnothing \}. \quad (18)$$

The correctness of this claim follows from observing that setting $z_{ij} = 1$ for all $(i,j) \in A$ will not change the value of the objective function. Thus the only cuts of interest in the above model are the ones that do not contain arcs in $\tilde{A}$. In terms of the natural variables the cut sets defined by (18) correspond to the (r,d) *separating nodecuts* in H, i.e., to the subsets of nodes $W_S \subseteq S$, such that for at least one NDC node pair (r,d), eliminating nodes from W_S and H results in a graph in which the nodes r and d are disconnected. Let $\mathcal{N}$ denote the union of all (r,d) separating node cuts in H, for all $r \in V_O$, $d \in V_D$. We can then rewrite the SCUT model for the GRLP in terms of the natural variables (where $z_{i'i''}$ is replaced by the node variables y_i, for all $i \in S$) as follows:

$$\text{(NSCUT)} \quad \min \sum_{i \in S} y_i$$

$$\text{s.t.} \sum_{i \in W_S} y_i \ge 1, \quad \forall W_S \in \mathcal{N}, \quad (19)$$

$$y_i \in \{0,1\} \quad \forall i \in S. \quad (20)$$

Note that it suffices to consider minimal separating node cuts in H to define the model NSCUT.

3.4. Branch-and-Cut Algorithm

For solving the GRLP to optimality, among all models presented previously, we chose the NSCUT model. As noted, NSCUT is essentially a strengthened formulation in the natural space of node variables. Since it has significantly fewer variables than either $\mathrm{DCUT}_{x,y}$ or MCF_y^+ (which are equally strong) it is better suited for computational purposes (in terms of solving large instances). Although it has an exponential number of constraints, the separation problem is fairly easy to solve. We now discuss the main algorithmic issues of an efficient implementation of the branch-and-cut framework for solving the NSCUT formulation.

3.4.1. Constructing the DSFP. When transforming a GRLP instance into an instance of the directed Steiner forest problem, we need to decide how to set up the set D of ordered NDC node pairs so that the number of arborescences that need to be simultaneously constructed, is minimized. To do so, we proceed in a greedy fashion as shown in Algorithm 1.

```
Algorithm 1 (OrderingNDCPairs(NDC))
  $D = \varnothing$;
  repeat
    Let $r \in T$ be the node with the highest
      frequency in NDC node pairs;
    for each $(r, i)$ or $(i, r)$ in NDC do
      $V_O = V_O \cup \{r\}$, $V_D = V_D \cup \{i\}$, $D = D \cup \{(r, i)\}$;
      NDC = NDC $\setminus \{(r, i), (i, r)\}$;
  until NDC is empty;
  return $D$.
```

3.4.2. Initialization. To speed up the performance of the branch-and-cut algorithm, one might consider the following set of inequalities in the initialization phase.

In-degree and Out-degree Inequalities: The following *in-degree inequalities* state that among all adjacent s-nodes of a destination node $k \in V_D$, at least one needs to be passed through: $\sum_{i:(i,k) \in A_H} y_i \ge 1$, for all $k \in V_D$. Similarly, the following *out-degree inequalities* ensure that among all adjacent s-nodes of a source node $r \in V_O$, at least one needs to be passed through: $\sum_{i:(r,i)\in A_H} y_i \geq 1$, for all $r \in V_O$. Both groups of inequalities are special cases of node-cut inequalities (19).

Flow-Balance Inequalities: Denote by S^{in} the set of all *internal s-nodes*, i.e., s-nodes that are not adjacent to a t-node. Observe that if a solution contains an s-node from S^{in}, then there must be some other s-node adjacent to V_D in the solution. Similarly, there must also be some other s-node adjacent to V_O in the solution. To state this in the enhanced directed graph $\tilde{H}$, we use the following inequalities: $y_i \leq \sum_{k:(i,k)\in A_H} y_k$ and $y_i \leq \sum_{k:(k,i)\in A_H} y_k$, for all $i \in S^{\text{in}}$.

We found that, although some instances were solved faster, on average the first group of constraints slowed down the performance of the branch-and-cut procedure. For the instances considered in our computational work the second group of constraints generally did not apply (i.e., the set S^{in} was typically empty). Therefore, in the computational results presented in §6, we did not add any cuts in the initialization phase except the trivial ones ($0 \leq y_i \leq 1$, for all $i \in S$).

3.4.3. Separation. The cut-set inequalities (19) can be separated in polynomial time. The separation algorithm relies on the calculation of multiple maximum flows on the *extended directed graph* $\tilde{H}$. Here, we explain an efficient way to do that. Given a fractional solution $\mathbf{y}'$ of the current LP-relaxation, we define the capacities on the arcs of the *support graph* as follows:

$$\text{cap}_{uv} = \begin{cases} y'_i, & (u,v) = (i', i''),\, i \in S; \\ 1, & \text{otherwise.} \end{cases}$$

This way, only the cuts from the collection $\mathcal{W}$ (on $\tilde{H}$) defined earlier (which correspond to node cuts for the collection $\mathcal{N}$ on H) will be separated. Before running the separation, the nodes of the set V_O are sorted into decreasing order according to the number of terminals of the arborescence rooted at r ($|V_D(r)|$), and a pool of node cuts is built. Using nested and back-cuts (as described in e.g., Ljubić and Gollowitzer 2013) up to 100 cuts are added into this pool, and then inserted into the master LP whose value is then recalculated. Detected violated node cuts are inserted as global cuts into the model, in every node of the branch-and-cut tree. Finally, our separation algorithm makes use of minimum-cardinality cuts (see, e.g., Ljubić and Gollowitzer 2013); i.e., during the separation procedure, among all node cuts with equal violation, those with the smallest cardinality are preferred.

4. Heuristic Approach

As the exact approach presented earlier is unable to solve large-scale GRLP instances, in this section we consider heuristic approaches. We propose two construction heuristics, referred to as GH1 and GH2 for the GRLP. We then explain a local search procedure, called *p-for-q*, in which p regenerator locations are replaced by q new ones ($p > q \geq 1$). The overall heuristic framework then works as follows: (1) preprocessing; (2) GH1 or GH2; (3) post-optimization (local search procedure).

4.1. Heuristic GH1

Our first heuristic GH1 focuses on S-components. At each iteration we identify a node $t \in T$ with the lowest T-degree (ties are broken randomly). We then examine all the neighboring S-components S_j that are connected to t. In each of the components S_j, we determine a set of nodes $L_j \subseteq S_j$ on which the regenerators are to be placed. Algorithm 2 provides the pseudocode of the procedure PLACE_REGENS that identifies L_j for a given component S_j and node t. We illustrate the algorithm in two examples given next. We assign weight $w(L_j)$ to each L_j by taking the ratio of the number of NDC node pairs that can be reduced after we deploy a regenerator at every node in L_j and the size of L_j. Let $L_t^{\max} \subseteq S_j$ denote the subset with the highest weight among all L_j's. GH1 deploys a regenerator at every node in $L_t^{\max}$. We then update the graph B (with UPDATE($B, L_t^{\max}, Q = \{1, 2, \ldots, n_S\}$)), and repeat the whole process while there exist NDC node pairs in B.

We first use a simple example given in Figure 5 to illustrate basic ideas of the heuristic. Initially all t-nodes have T-degree equal to zero. GH1 arbitrarily chooses node t_1. Node t_1 is connected to three S-components: $S_1 = \{s_6\}$, $S_2 = \{s_1, s_2, s_3\}$, and $S_3 = \{s_7\}$. If we place regenerators at all nodes from S_j (i.e., $L_j = S_j$, $1 \leq j \leq 3$), the weights for L_1, L_2, and L_3 will be equal to 1, 10/3, and 1, respectively. GH1 sets $L_{t_1}^{\max} = L_2$ and places a regenerator at each of the nodes s_1, s_2, and s_3. In this case, after we update the graph, we obtain a feasible solution and the algorithm stops.

In general, it is not necessary to place regenerators at all s-nodes within a component S_j and our procedure PLACE_REGENS takes this into account. We first

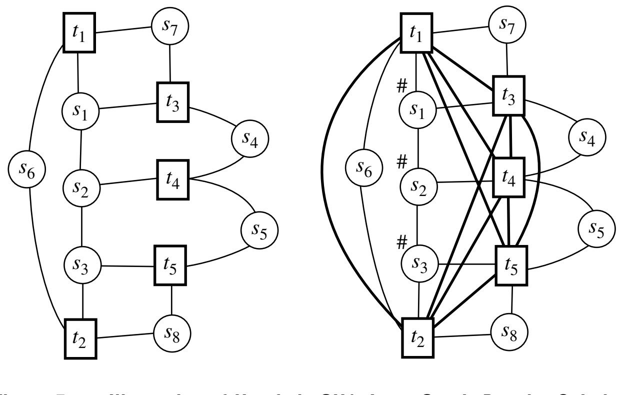


Figure 5 **Illustration of Heuristic GH1: Input Graph *B* and a Solution Found by GH1**

select an arbitrary s-node $s^* \in S_j$ adjacent to t and perform a *restricted breadth-first-search* (BFS) (denoted by RESTRICTED_BFS in Algorithm 2) in which the BFS is performed on the subgraph of B induced by the set of nodes $S_j \cup T$ with s^* being a root. The search in one direction is interrupted each time a t-node is encountered. That way, all t-nodes adjacent to the given S_j are leaves of the resulting BFS tree that we denote by $T_{\text{BFS}} = (V_{\text{BFS}}, E_{\text{BFS}})$. In a subsequent iterative process, we eliminate s-nodes that are leaves of that tree. Observe at this point that if regenerators are placed at all the internal nodes of T_{BFS}, all t-nodes adjacent to the given S_j can communicate. Rather than doing so, we try to rearrange the edges in T_{BFS}, so that the number of its internal s-nodes is minimized. The idea is to try to reduce the number of internal s-nodes in T_{BFS} without sacrificing the number of NDC node pairs that can be reduced after a regenerator is added to each one of these internal nodes. At each iteration the number of s-nodes on the longest branch (from s^*) in T_{BFS} represents the minimum number of regenerators that need to be deployed before the node t can communicate with a leaf node $l \in T$ via S_j (this is because breadth-first-search is used to construct T_{BFS}). We try to make the best use of the longest branch by merging it with other branches (or parts of these branches). We then eliminate parts of the truncated branches from T_{BFS} consisting of s-nodes that are not used to connect NDC nodes pairs. Finally, the set L_j corresponds to the set of all internal s-nodes of the resulting tree T_{BFS}.

```
**Algorithm 2** (PLACE_REGENS($S_j$, $t$))
  Arbitrarily select a neighboring $s$-node of $t$, say
    $s^* \in S_j$;
  $T_{\text{BFS}} = (V_{\text{BFS}}, E_{\text{BFS}})$ = RESTRICTED_BFS($S_j \cup T$, $s^*$);
  Mark all $s$-nodes from $V_{\text{BFS}} \cap S_j$ as "unmerged";
  Iteratively eliminate from $T_{\text{BFS}}$ $s$-nodes that are
    leaves, until there are no such nodes left;
  **while** $\exists s \in S_j \cap V_{\text{BFS}}$ labeled as "unmerged" **do**
    Select a longest (from $s^*$) "unmerged" branch
      $\mathcal{B}$ in $T_{\text{BFS}}$;
    **for** each $s$-node $l \in \mathcal{B}$ **do**
      **for** each $e = \{k, l\} \in E$ **do**
        **if** $k \in V_{\text{BFS}}$ and $e \notin E_{\text{BFS}}$ **then**
          Delete the edge from $k$ to its predecessor
            in $E_{\text{BFS}}$;
          $E_{\text{BFS}} = E_{\text{BFS}} \cup \{e\}$;
      Mark $l$ as "merged";
    Iteratively eliminate from $V_{\text{BFS}}$ $s$-nodes that are
      leaves, until there are no such nodes left;
  **return** $L_j = V_{\text{BFS}} \cap S_j$.
```

Figure 6 illustrates the procedure PLACE_REGENS for given $S_1 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ and the identified t-node t_1 and $s^* = s_1$. The BFS tree T_{BFS} is constructed through Figures 6(a)–(c). The merging process is shown in Figures 6(d)–(f). The longest unmerged branch from s_1 consists of $\{s_1, s_4, s_7, t_3\}$ (ties are broken randomly). While scanning this path, we delete edges $\{s_2, t_2\}$ and $\{s_3, s_6\}$ and replace them with $\{s_7, t_2\}$ and $\{s_7, s_6\}$, respectively. Nodes s_2 and s_3 become leaves and after their deletion, we obtain a tree T_{BFS} such that

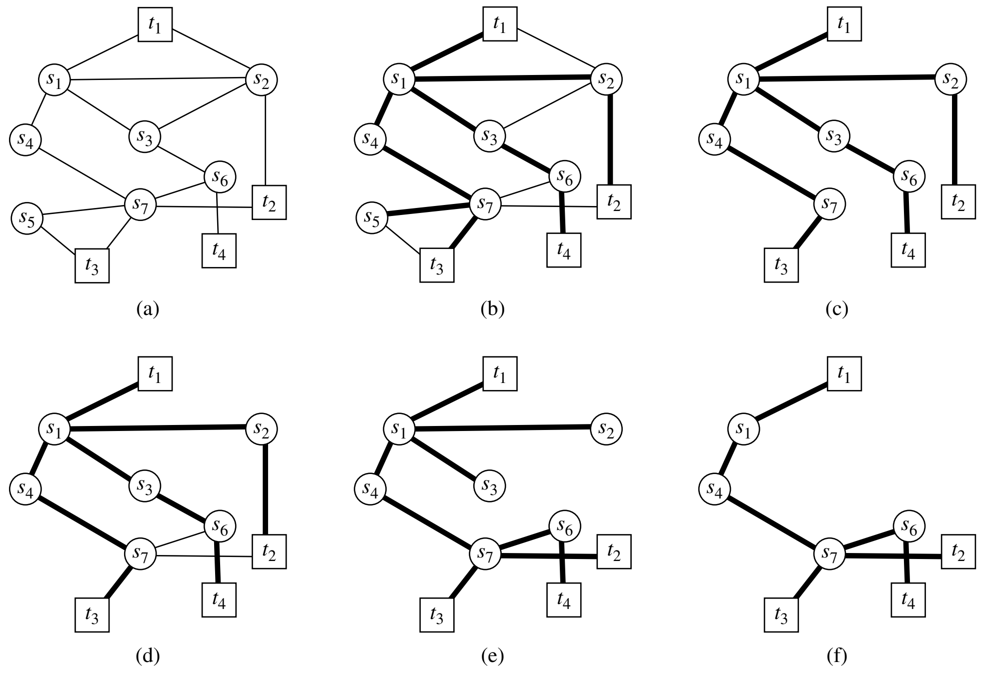


Figure 6 Illustration of the RESTRICTED_BFS ((a)–(c)) and PLACE_REGENS ((d)–(f)) Procedure

$L_1 = \{s_1, s_4, s_6, s_7\}$. Since six NDC node pairs—(t_1, t_2), (t_1, t_3), (t_1, t_4), (t_2, t_3), (t_2, t_4), and (t_3, t_4)—can communicate after a regenerator is placed at each of the nodes from L_1, we have $w(L_1) = 6/4 = 1.5$.

Lemma 7. *Heuristic GH1 runs in $O(|T||S||N|^2)$ time.*

Proof. See online supplement. □

4.2. Heuristic GH2

Heuristic GH2 is a somewhat greedier version of the heuristic GH1. The main difference between GH1 and GH2 lies in the way we place regenerators, once the BFS tree T_{BFS} is found. As opposed to GH1, GH2 focuses on placing regenerators only on the longest branch in T_{BFS}. For each neighboring S-component S_j of a selected node $t \in T$, in the corresponding BFS tree we identify the longest branch. Let B_j denote the set of internal nodes in the identified branch. GH2 then computes $w(B_j)$ by taking the ratio of the number of NDC node pairs that can be reduced after a regenerator is placed at every node in B_j and the size of B_j. The branch $B_t^{\max}$ with the largest $w(B_j)$ value among all neighboring S-components of t is then selected for deploying regenerators. The graph B is then updated (using the UPDATE-procedure), a new t-node with the lowest degree is selected, and the whole procedure is repeated until all NDC node pairs can communicate with each other. We note that the running times of the heuristic GH2 remain unchanged from the modifications it makes on GH1.

4.3. Local Search Procedure

The post-optimization local search (POLS) procedure consists of a basic subroutine called *p-for-q*. POLS is applied to the solutions obtained from GH1 or GH2 to improve upon them. Subroutine *p-for-q* is a local search procedure that tries to replace p regenerator locations in the current solution with q new regenerator locations. When *p-for-q* results in a feasible solution, its application reduces the number of regenerators in the solution by $(p - q)$. We consider two variants of this procedure: (1) *p-for-q* replacement within the same S-component, and (2) *p-for-q* replacement between two different S-components. In the first case, for a given S-component S_i, and a set of regenerators placed in $L_i \subset S_i$, we try to replace p regenerator locations $P \subseteq L_i$ by q new ones, $Q \subseteq S_i \backslash L_i$. In the second case, for two disjoint S-components (S_i, S_j) and two sets of locations where regenerators are placed, $L_i \subseteq S_i$ and $L_j \subset S_j$, we try to replace p regenerator locations $P \subseteq L_i$ with q new locations, $Q \subseteq S_j \backslash L_j$.

Given a feasible solution $L \subseteq S$, we iteratively apply *p-for-q* moves until we end up with a solution that cannot be improved anymore (our local search selects the first improvement). Then we change the neighborhood by switching to new values for p and q and repeat the same process again. We implement 2-for-1, 3-for-2, and 4-for-3 improvement, within each S-component, and 2-for-1 improvement between two S-components.

Lemma 8. *Each single* p-for-q *move takes at most $O(|S||N|^2)$ time.*

Proof. See online supplement. □

The number of possible *p-for-q* moves is bounded by $|S|^{p+q}$. In case the moves are performed within the same S-component, the search of the *p-for-q* neighborhood takes at most $\sum_{i=1}^{n_S} O(|S|^{p+q}|S||N|^2) = O(|S|^{p+q+2}|N|^2)$ time. Similarly, if the moves are performed between two distinct components, the search will take at most $O(|S|^{p+q+3}|N|^2)$ time (this is because for each fixed S-component from which p nodes are to be replaced, there are $O(|S|)$ possibilities to select the other component for the exchange).

5. Weighted GRLP

The weighted version of the GRLP (WGRLP) is motivated by recognizing that in many real-world situations the cost of installing regenerators at different nodes of a network may vary due to real estate costs. In particular, installing and maintaining a regenerator at a dense urban area may be much more expensive than in a small town (or a rural area). Mathematically, the WGRLP is identical to the GRLP, except that the objective is to minimize the weighted sum of nodes selected as regenerator locations. If we let w_i denote the weight of node i, the objective of the WGRLP is to minimize $\sum_{i \in L} w_i$, while the objective of the GRLP is to minimize $\sum_{i \in L} 1 = |L|$.

The construction of the communication graph B and the preprocessing procedure described in §2 do not depend on weights and thus remain unchanged for the WGRLP. Regarding the MIP formulations proposed in §3, we only need to change the objective function to address the WGRLP. With respect to heuristics, we only need to make minor modifications. More precisely, for both of the heuristics we define the *longest branch* as the one with the highest total weight of all its internal unmerged nodes. That way, we always select the branch that maximizes the number of NDC node pairs eliminated *per unit cost*. Finally, it is straightforward to incorporate weights into the POLS procedure and its running time per single *p-for-q* move remains unchanged. In addition to the moves implemented for the GRLP, we also allow the following four moves if they lead to a feasible solution with reduced total weight: 3-*for*-3, 2-*for*-2, and 1-*for*-1 within each S-component, and 2-*for*-2* between S-components where two regenerator locations in one S-component are replaced by one regenerator location from the same S-component and one from another S-component.

6. Computational Results

For solving the linear programming relaxations and for a generic implementation of the branch-and-cut approach, we used the commercial packages IBM CPLEX (version 12.5) and Concert Technology (CPLEX 2012). Our experiments were performed on an Intel Core i7 (2600) 3.4 GHz machine with 16 GB RAM, where each run was performed on a single processor. All implementations are done using C++ programming language. For calculating the maximum flow we adapted the implementation of Cherkassky and Goldberg (1997).

6.1. Data Sets

In our study on the RLP in Chen et al. (2010), we generated three types of networks: (i) randomly generated networks in which the communication graph was generated explicitly; (ii) networks with random distances; and (iii) Euclidean networks. We found that the instances of the first group (where the communication graph was generated directly) were much harder to solve than the other two types of networks. Consequently, in our computational study on the GRLP, we focus on randomly generated communication graphs B. The instance generator procedure described next guarantees that the sets S and T are disjoint and that the instances are feasible.

Set 1 (*GRLP Instances*): Each instance of this set is generated according to two parameters: the first one, n, controls the number of nodes ($|N|$), and the other one, $p \in [0,1]$, determines the percentage of s-nodes among them. The number of S-components, n_S, is a random integer value chosen from $\{2,\dots,5\}$ with equal probability. Each s-node is randomly assigned to one of the S-components with probability $1/n_S$. Notice that the probability of a particular S-component being empty is $((n_S-1)/n_S)^{|S|}$. When this happens, we move a randomly chosen s-node into the empty S-component. This makes sure that there are no empty S-components. We define the density of an S-component as $d_i = 2|E_i|/(|S_i|(|S_i|-1))$, where $|E_i|$ is the number of edges in the S-component S_i and $|S_i|$ its number of nodes. Each S_i has a 50-50 chance of getting assigned either a high ($d_i = 70\%$) or low density ($d_i = 30\%$). To ensure connectivity of each S-component S_i, we first generate a tree spanning all its nodes and then randomly add edges until its density reaches the assigned level d_i. Each t-node ℓ and an S-component S_j are connected according to the parameter q whose value is randomly chosen from the set $\{0, 1/|S_j|, 2/|S_j|, \dots, (|S_j|-1)/|S_j|\}$ with equal probability for each element. We then randomly generate $q \times |S_j|$ distinct edges between ℓ and S_j. Observe that the probability of a t-node ℓ being isolated is $\Pi_{j=1}^{n_S}(1/|S_j|)$. When this happens, we arbitrarily choose a component S_i and randomly select $q \in \{1/|S_i|, 2/|S_i|, \dots, (|S_i|-1)/|S_i|\}$ with equal probability given to each element. This makes sure that every t-node is connected to at least one s-node. In the resulting graph, we check every pair of t-nodes: only if they share at least one common S-component do we add them to the set of NDC node pairs. This final step guarantees that all generated instances are feasible. Set1 contains 10 instances for each $n \in \{50, 75, 100, 125, 150\}$ and each $p \in \{0.25, 0.5, 0.75\}$.

Set 2 (*WGRLP Instances*): We generate instances for WGRLP using the same underlying graphs as in Set1. We only modify these graphs by assigning to each s-node an integer weight randomly chosen from $\{2,3,4\}$ with equal probability.

Set 3 (*Large-scale GRLP Instances*): This set of instances is generated following the same rules as for the Set1. Set3 contains 10 instances for each $n \in \{175, 200, 300, 400, 500\}$ and each $p \in \{0.25, 0.5, 0.75\}$.

6.2. Results

For the sets of generated benchmark instances, we now report on the results obtained by running the branch-and-cut algorithm (B&C) described in §3, and by running the two heuristic frameworks described in §4 (preprocessing, construction heuristic and local search), that we will refer to as GH1-Framework and GH2-Framework. We will use the B&C algorithm to measure the quality of heuristic solutions, by comparing the corresponding solution values. B&C was given a time limit of one hour.

Tables 1, 2, and 4 summarize the computational results for Set1, Set2, and Set3, respectively. Each row of these three tables aggregates results from 10 instances with the same parameter settings. The parameter settings are specified in the first two columns "n" and "p(%)." Column "#red" in Table 1 reports on the average number of nodes eliminated in the preprocessing procedure. (Since the graphs for Set 2 are identical to Set 1 and thus the preprocessing identical, we do not repeat this information in Table 2. Because preprocessing did not reduce the size of any of the large instances in Set 3, we do not include this column in Table 4.) Blocks "GH1-framework" and "GH2-framework" report the computational results for the two heuristic frameworks, respectively. In particular, "NR" is the average number of regenerators obtained by the heuristic framework; "RT" reports the average running time in seconds; and "#opt" reports the number of optimal solutions obtained by the heuristic framework (in the case where the optimum is known). In columns denoted by "#imp" we report the number of times the post-optimization local search improved the solution found by the underlying construction heuristic. Column "#PO" denotes the average number of regenerators that were reduced after applying the POLS procedure. For the B&C algorithm we report the following values: lower and upper bounds ("LB" and "UB," respectively) and the running time ("RT") averaged over 10 instances per group. The "#opt" column shows how many (of the 10) instances B&C solved to optimality. For the set of the WGRLP instances (Set 2), instead of "NR," we report "Obj" values, i.e., the objective function values averaged over 10 instances per group. When the B&C algorithm terminates without solving the problem to optimality, we round up the fractional lower bound. If the heuristic solution equals this lower bound, we state that the heuristic has found the optimal solution. To demonstrate the quality of the NSCUT formulation, column LB_r shown in Tables 1 and 2 reports the LP-relaxation values obtained at the root node.

Set 1. Of 150 instances of this group, B&C solves all but one to provable optimality within the time limit of one hour. Regarding the overall performance of the B&C approach, we observe that, for a fixed number of nodes n, instances with $p = 0.5$ (the percentage of s-nodes in B) appear to be more challenging than the remaining ones. This can be explained by the fact that the computational complexity of the exact approach is directly proportional to the two values: (a) the number of NDC node pairs and (b) the number of s-nodes in the input graph. Increasing the values of p (i.e., setting $p = 0.75$) implies fewer NDC node pairs, and therefore, fewer cutting planes that need to be inserted before the optimal solution is found. On the other hand, lowering the values of p (i.e., setting $p = 0.25$) implies lower percentage of s-nodes, and therefore, the reduction of the size of the search space (as there are fewer z variables involved).

Comparing the two heuristic frameworks, from Table 1, we conclude that the GH1-Framework is marginally better than the GH2-Framework in terms of the number of times it finds the best solution. Notice that the two heuristic frameworks have approximately the same running times. Looking closer at the difference between the better among the two heuristic frameworks and the optimal solution value (or, the lower bound in one remaining case), we observed that in 134 cases optimal solutions were found, for 13 instances, this difference was equal to one, and for the remaining three cases, this difference was equal to two. Both heuristic approaches solved each single Set 1 instance within a few seconds, whereas the running times of the branch-and-cut algorithm exponentially increase with the number of nodes and NDC node pairs. Of the 150 instances, preprocessing helped in 83. Interestingly, no regenerators were fixed in the preprocessing.

Comparing the two heuristic frameworks we find that the GH1-Framework provides the best solution in 146 instances, finds the optimal solution in 132 instances, and POLS was useful (i.e., found an improvement to the greedy solution) in 143 instances. The GH2-Framework provides the best solution in 144 instances, finds the optimal solution in 128 instances, and POLS was useful in 141 instances. The GH1-Framework solely provides the best heuristic solution in six instances, whereas the GH2-Framework solely provides the best heuristic solution in four instances. Hence, there are always instances where each one of the heuristics solely provides the best solution. We also compare the performance of the construction heuristics GH1 and GH2, before applying the POLS. The values provided in the column denoted by "#PO" indicate that the POLS was slightly more effective for GH1, indicating that the values of GH1 were slightly worse than those obtained by GH2. We may also conclude that the POLS plays a significant role in the performance of the proposed heuristic frameworks.

Table 1 Computational Results for Set 1

			GH1-framework					GH2-framework						B&C			
n	p (%)	#red	NR	RT	#opt	#PO	#imp	NR	RT	#opt	#PO	#imp	UB	LB	LB_r	RT	#opt
50	25	2.2	7.2	0.0	10	1.9	9	7.2	0.0	10	1.5	9	7.2	7.2	7.2	0.2	10
	50	0	7.7	0.0	8	2.0	7	7.7	0.0	8	1.6	7	7.5	7.5	7.4	0.2	10
	75	0.9	4.3	0.0	10	2.7	8	4.3	0.0	10	2.1	7	4.3	4.3	4.1	0.1	10
75	25	0.3	9.8	0.0	9	1.8	10	9.8	0.0	9	1.2	9	9.7	9.7	9.5	1.4	10
	50	1.4	9.6	0.0	7	5.0	10	9.5	0.0	8	3.8	10	9.2	9.2	8.3	2.2	10
	75	9.7	5.5	0.0	10	3.6	9	5.6	0.0	9	2.9	9	5.5	5.5	4.6	0.6	10
100	25	8.3	11.8	0.0	9	3.1	10	11.8	0.0	9	1.7	10	11.7	11.7	11.0	6.7	10
	50	3	10.9	0.0	9	6.5	10	11	0.0	8	3.9	10	10.8	10.8	9.0	21.5	10
	75	0	7.2	0.0	8	4.3	10	7.3	0.0	7	3.1	10	7.0	7.0	6.1	3.2	10
125	25	0	14.1	0.0	10	3.9	10	14.1	0.0	10	3.1	10	14.1	14.1	12.7	31.7	10
	50	0.9	12.1	0.2	9	6.5	10	12	0.1	9	5.1	10	11.9	11.9	9.5	237.1	10
	75	1.7	8.4	0.0	8	5.9	10	8.5	0.1	6	5.4	10	8.1	8.1	6.8	14.2	10
150	25	0.5	14.5	0.1	10	4.4	10	14.5	0.1	10	4.1	10	14.5	14.5	12.2	259.8	10
	50	11.6	12.3	0.4	8	9.4	10	12.3	0.5	8	6.9	10	12.2	12.0	9.5	838.6	9
	75	16.9	9.5	0.1	7	6.5	10	9.5	0.2	7	4.9	10	9.1	9.1	7.3	403.4	10

Table 2 Computational Results for Weighted Instances (Set 2)

		GH1-Framework					GH2-Framework					B&C				
n	p (%)	obj	RT	#opt	#PO	#imp	obj	RT	#opt	#PO	#imp	UB	LB	LB_r	RT	#opt
50	25	21.4	0.0	9	6.8	10	21.4	0.0	9	5.5	10	21.3	21.3	20.9	0.2	10
	50	21.4	0.0	9	9.2	10	21.3	0.0	10	7.1	10	21.3	21.3	21.1	0.2	10
	75	11.5	0.0	8	11.5	10	11.5	0.0	8	8.3	9	11.3	11.3	11.0	0.1	10
75	25	28.4	0.0	9	6.6	10	28.4	0.0	9	5.4	10	28.2	28.2	27.4	1.2	10
	50	26.1	0.0	8	16.5	10	25.8	0.0	9	15.9	10	25.7	25.7	23.4	2.2	10
	75	14.5	0.0	8	12.8	10	14.2	0.0	8	9.8	10	13.9	13.9	13.9	0.4	10
100	25	35.1	0.0	9	10.9	10	35.0	0.0	10	6.9	10	35.0	35.0	33.4	5.2	10
	50	28.8	0.0	9	24.1	10	28.9	0.2	9	16.0	10	28.6	28.6	25.4	15.9	10
	75	19.1	0.0	6	14.6	10	18.9	0.1	8	14.3	10	18.7	18.7	16.1	2.3	10
125	25	42.6	0.1	10	16.7	10	42.6	0.1	10	13.3	10	42.6	42.6	38.5	35.1	10
	50	32.5	0.2	7	24.8	10	32.3	0.5	8	18.1	10	31.6	31.6	26.4	124.7	10
	75	20.8	0.2	5	22.4	10	20.9	0.0	7	20.2	10	20.1	20.1	17.3	9.6	10
150	25	42.0	0.1	9	19.4	10	42.1	0.3	9	15.5	10	41.9	41.9	37.2	146.1	10
	50	33.3	1.2	7	33.3	10	32.8	0.6	8	26.3	10	32.3	32.3	27.1	562.3	10
	75	22.1	0.2	7	24.3	10	21.9	0.5	9	21.8	10	21.4	21.4	18.8	31.2	10

Set 2. Table 2 reports computational results for Set 2. The B&C procedure solves all 150 instances to optimality, and needs, on average, less time than solving the same instances of Set 1. This can be explained by the fact that WGRLP solutions are less symmetric than the corresponding unweighted ones. The better of the two heuristic frameworks finds optimal solution in 134 cases; in eight cases the difference between the heuristic and the optimal solution is equal to one; and in the remaining eight cases this difference varies between 2 and 5. Comparing the two heuristic frameworks we find that the GH1-Framework provides the best solution in 135 instances, finds the optimal solution in 120 instances, and POLS was useful in all instances. The GH2-Framework provides the best solution in 145 instances, finds the optimal solution in 131 instances, and POLS was useful in all but one instance. In five instances the GH1-Framework was solely the best heuristic approach, whereas the GH2-Framework was solely the best heuristic approach in 15 instances. Hence, we conclude that the GH2-Framework performs marginally better than the GH1-Framework. Notice that, as for Set 1, the running times of the two frameworks are about the same.

Table 3 compares the number of branch-and-bound nodes (BBnodes) in the B&C procedure on Sets 1 and 2. When BBnodes $= 0$, it indicates that the problem was solved to optimality at the root node. Table 3 shows that as the number of nodes increases, the number of BBnodes increases. For input graphs with more than 75 nodes, problems are seldom solved at the root node. Furthermore, the number of BBnodes significantly increases as p gets closer to 50%.

Set 3. Table 4 shows the corresponding results for the large-scale instances (Set 3). Since the running time of the POLS increases with the number of nodes, for graphs with $n \geq 300$ we used only 2-*for*-1 local search. Of 150 instances, the branch-and-cut procedure finds optimal solutions only in 47 cases. The value of the best found heuristic solution is given as the upper cutoff tolerance to the branch and cut; i.e., CPLEX cuts off or discards any solutions that are greater than the specified upper cutoff value. This way, five heuristic solutions were indirectly proved to be optimal. Of these 47 instances, the better of the two heuristic solutions is optimal in 39 cases. Comparing the two heuristic approaches we find that the GH1-Framework provides the best solution in 142 instances, finds the optimal solution in 39 instances, and POLS was useful in all instances. The GH2-Framework provides the best solution in 130 instances, finds the optimal solution in 34 instances, and POLS was useful in all instances. In 20 instances the GH1-Framework was solely the best heuristic approach, whereas the GH2-Framework was solely the best heuristic approach

Table 3 Number of Branch-and-Bound Nodes (Set 1 and Set 2)

		Set 1			Set 2		
n	p (%)	Median	Mean	BBnodes $= 0$	Median	Mean	BBnodes $= 0$
50	25	0.5	1.1	5	0	1.1	6
50	50	2.5	2.5	2	2.5	2.8	2
50	75	1	2.4	4	1	1.667	4
75	25	3	3.1	2	2	2.2	3
75	50	3.5	10.2	3	4	11.1	0
75	75	7	8.5	1	5	6	0
100	25	6	6.7	2	5	5.7	0
100	50	41	47.2	1	18.5	41.1	0
100	75	8	9.8	0	7.5	10.1	0
125	25	13	14.9	1	17	22.1	0
125	50	128	485	0	165.5	186.9	0
125	75	13	33.3	2	20	21.8	0
150	25	16.5	214.6	0	30	71	0
150	50	301	522.1	0	154	433	0
150	75	111	1,427	0	16.5	43.6	0

Table 4 Computational Results for Large Scale Instances (Set 3)

		GH1-framework					GH2-framework					B&C			
n	p (%)	NR	RT	#opt	#PO	#imp	NR	RT	#opt	#PO	#imp	UB	LB	RT	#opt
175	25	16.4	0.1	8	7.3	10	16.5	0.3	8	4.8	10	16.3	16.2	1,625.5	9
	50	13.2	1.0	4	8.7	10	13.4	1.0	2	7.8	10	13.1	12.3	2,667.0	5
	75	9.7	0.4	8	7.7	10	9.6	0.4	7	7.0	10	9.1	9.1	245.3	10
200	25	17.4	0.3	5	7.5	10	17.5	0.4	4	4.4	10	17.4	16.2	2,869.0	5
	50	13.0	2.6	3	9.9	10	13.0	2.8	3	8.1	10	12.8	11.3	3,142.0	4
	75	10.6	1.7	7	9.9	10	10.5	1.7	7	7.6	10	10.0	10.0	1,116.0	10
300	25	16.8	1.0	2	7.9	10	17.4	1.3	2	5.5	10	16.8	15.3	3,600.0	2
	50	15.4	1.2	1	9.4	10	16.1	2.4	1	6.5	10	15.3	12.6	3,600.0	1
	75	12.5	1.5	1	8.4	10	12.9	2.1	0	6.7	10	12.4	9.9	3,600.0	1
400	25	22.4	7.4	0	9.6	10	22.4	8.9	0	7.0	10	21.9	13.3	3,600.0	0
	50	19.8	20.3	0	11.4	10	20.0	18.8	0	10.4	10	19.6	11.1	3,600.0	0
	75	14.7	4.7	0	11.6	10	14.4	6.0	0	10.2	10	14.4	9.5	3,600.0	0
500	25	25.0	21.6	0	11.3	10	24.8	28.4	0	9.5	10	24.8	12.1	3,600.0	0
	50	21.4	20.3	0	13.5	10	21.4	33.2	0	11.7	10	21.4	8.8	3,600.0	0
	75	12.7	32.4	0	8.4	10	15.0	28.2	0	9.0	10	12.7	7.6	3,600.0	0

in eight instances; i.e., there are always instances where each one of the approaches solely provides the best solution. However, the GH1-Framework performs marginally better than the GH2-Framework. Notice that, as for the previous two instance groups, the running times of the two heuristic frameworks are about the same.

7. Conclusions

In this paper, we introduced and addressed the generalized regenerator location problem, which is an important and fundamental problem that arises in the design of fiber optic networks. We developed a graph transformation procedure that simplifies conceptualization of the problem. We also addressed the node-weighted version of the problem (WGRLP) that allows for modeling location-dependent regenerator costs. Several safe preprocessing procedures are introduced.

Although the RLP has been studied previously, the procedures developed to solve it do not apply to the GRLP. We show that the GRLP can be transformed into a particular instance of the NWDSFP on a directed graph. Using this transformation, we developed a natural formulation and two extended formulations for the GRLP. We then showed how to strengthen these three formulations by disaggregation in one extended formulation and by node-splitting in the natural formulation and the flow-based extended formulation. We proved the equivalence of these three strengthened formulations, and developed a branch-and-cut procedure for the strengthened formulation on the natural node selection variables (NSCUT). We also proposed two construction heuristics and a local search procedure (a post-optimizer) that tries to reduce the cost of the solution by replacing p regenerator locations by q new ones ($p > q \geq 1$).

Our computational study was conducted on a set of 150 unweighted (Set 1) and 150 weighted (Set 2) instances with between 50 and 150 nodes and with different densities. Additionally, we tested 150 larger graphs (Set 3) with between 175 and 500 nodes. The results obtained indicate that in both the weighted and unweighted case the quality of the heuristic solutions is quite good. While the running time of the heuristics showed a minor increase with the increase of the size of input graphs, the branch-and-cut algorithm showed certain limitations. Due to the exponential increase of the running time of the branch-and-cut algorithm, at present, for 104 of 450 instances, optimal solution values remain unknown. In that context, both heuristics, whose running times remain less than one minute even for the largest test instances, appear to be a viable approach for real-world problems dealing with large-scale instances.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2014.0621.

Acknowledgments

The authors thank the referees and associate editor for their feedback that improved the presentation of this work. Support to I. Ljubić from the Austrian Program for Advanced Research and Technology Fellowship of the Austrian Academy of Sciences, and to S. Raghavan from the Smith School Strategic Research Fund are gratefully acknowledged.

References

Bathula B, Sinha R, Chiu A, Feuer M, Li G, Woodward S, Zhang W, Doverspike R, Magill P, Bergman K (2013) Cost optimization using regenerator site concentration and routing in ROADM networks. *9th Internat. Conf. Design of Reliable Comm. Networks, 2013 (DRCN 2013), Budapest, Hungary*, 139–147.

Borella MS, Jue JP, Banerjee D, Ramamurthy B, Mukherjee B (1997) Optical components for WDM lightwave networks. *Proc. IEEE* 85(8):1274–1307.

Chen S, Raghavan S (2007) The regenerator location problem. *Proc. Internat. Network Optim. Conf. (INOC 2007), Spa, Belgium*.

Chen S, Ljubić I, Raghavan S (2009) The generalized regenerator location problem. *Proc. Internat. Network Optim. Conf. (INOC 2009), Pisa, Italy*.

Chen S, Ljubić I, Raghavan S (2010) The regenerator location problem. *Networks* 55(3):205–220.

Cherkassky BV, Goldberg AV (1997) On implementing push-relabel method for the maximum flow problem. *Algorithmica* 19(4):390–410.

CPLEX (2012) Accessed March 10, 2012, http://www.ilog.com/products/cplex/.

Fernandes ML, Gouveia L (1998) Minimal spanning trees with a constraint on the number of leaves. *Eur. J. Oper. Res.* 104(1): 250–261.

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 Trans. Netw.* 19(2): 498–511.

Ford LR Jr, Fulkerson DR (1956) Maximal flow through a network. *Canadian J. Math.* 8:399–404.

Gouveia L, Patricio P, Sousa A, Valadas R (2003) MPLS over WDM network design with packet level QoS constraints based on ILP models. *Proc. IEEE Infocom*, Vol. 1 (IEEE Press, Piscataway, NJ), 576–586.

Ljubić I, Gollowitzer S (2013) Layered graph approaches to the hop constrained connected facility location problem. *INFORMS J. Comput.* 25(2):256–270.

Lucena A, Maculan N, Simonetti L (2010) Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Sci.* 7(3):289–311.

Lucerna D, Gatti N, Maier G, Pattavina A (2009) On the efficiency of a game theoretic approach to sparse regenerator placement in WDM networks. *GLOBECOM'09: Proc. 28th IEEE Conf. Global Telecommunications* (IEEE Press, Piscataway, NJ), 354–359.

Mertzios G, Sau I, Shalom M, Zaks S (2012) Placing regenerators in optical networks to satisfy multiple sets of requests. *IEEE/ACM Trans. Network* 20(6):1870–1879.

Mertzios G, Shalom M, Wong P, Zaks S (2011) Online regenerator placement. Fernández Anta A, Lipari G, Roy M, eds. *Proc. 15th Internat. Conf. Principles of Distributed Systems, LNCS 7109* (Springer, Berlin), 4–17.

Mukherjee B (2000) WDM optical communication networks: Progress and challenges. *IEEE J. Selected Areas Comm.* 18(10): 1810–1824.

Pachnicke S, Paschenda T, Krummrich P (2008) Assessment of a constraint-based routing algorithm for translucent 10 Gbits/s DWDM networks considering fiber nonlinearities. *J. Optical Networking* 7(4):365–377.

Patel A, Gao C, Jue JP, Wang X, Zhang Q, Palacharla P, Naito T (2010) Traffic grooming and regenerator placement in impairment-aware optical WDM networks. *Proc. 14th Conf. Optical Network Design Model* (IEEE Press, Piscataway, NJ), 72–77.

Rumley S, Gaumier C (2009) Cost aware design of translucent WDM transport networks. *Proc. 11th Internat. Conf. Transparent Optical Networks, Azores, Portugal*.

Sen A, Banerjee S, Ghosh P, Murthy S, Ngo H (2010) Brief announcement: On regenerator placement problems in optical networks. *Proc. 22nd ACM Sympos. Parallelism in Algorithms and Architectures*, SPAA '10 (ACM, New York), 178–180.

Yetginer E, Karasan E (2003) Regenerator placement and traffic engineering with restoration in GMPLS networks. *Photonic Network Comm.* 6(2):139–149.

Yildiz B, Karasan O (2015) Regenerator location problem and survivable extensions: A hub covering location perspective. *Transportation Res. B* 71:32–55.

Zymolka A (2006) Design of survivable optical networks by mathematical optimization. Ph.D. thesis, Technische Universität Berlin, Germany.