The Generalized Regenerator Location Problem

Si Chen* Ivana Ljubić† S. Raghavan‡

*College of Business and Public Affairs
Murray State University, Murray, KY 42071, USA
si.chen@murraystate.edu

†Department of Statistics and Operations Research
University of Vienna, Brünnerstr. 72, 1210 Vienna, Austria
ivana.ljubic@univie.ac.at

‡Robert H. Smith School of Business & Institute for Systems Research
University of Maryland, College Park, MD 20742, USA
raghavan@umd.edu

Abstract

In an optical network a signal can only travel a maximum distance of 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. 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 mixed-integer programming (MIP) models for the GRLP and compare the strength of these models. We then 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, while both heuristic frameworks are high-quality approaches for solving large-scale instances.

Keywords: optical network design; (node-weighted) directed Steiner forest; branch-and-cut; heuristics; minimum connected dominating set; regenerator location

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, 1999) 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, as the cost of regenerators is very high (for example, Mertzios et al., 2010, 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. $S \subseteq N$ is the set of candidate locations where regenerators can be placed, $T \subseteq N$ is the set of terminal nodes that must communicate with each other. A mapping $\ell : F \mapsto \mathbb{R}^+ \cup \{0\}$ 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 from N 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 problem 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., $T \neq N$). Further, 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, while 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 to 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 (Flammini et al., 2011; Lucerna et al., 2009; Mertzios et al., 2010; Pachnicke et al., 2008; Rumley and Gaumier, 2009), though 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 the node-weighted directed Steiner forest problem (NWDSFP). Unfortunately, typical multicommodity or directed cut formulations that are known to be extremely strong formulations for the Steiner tree problem are weak and are not computationally viable for the GRLP. Using a technique of disaggregation—which we show to be computationally equivalent to node splitting—results in a strong formulation for the GRLP. We then develop a branch-and-cut (B&C) approach based on the node splitting model to provide exact solutions for the GRLP. The node-splitting approach results in a model with fewer variables (while still being the strongest model introduced in this paper) and 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 heuristic and the local search procedure, is applied. These heuristic frameworks are extremely fast and produce high-quality solutions for the GRLP. Specifically, out of 450 instances studied in this paper (ranging from 50 to 500 nodes) the B&C approach solves 326 instances to optimality within the given time limit of one hour per instance. Of these 326 instances our heuristic frameworks found the optimal solution 290 times. Even for the largest 500 node problems, our heuristic frameworks typically run in about half a minute per instance while the B&C approach is unable to generate any meaningful upper bounds (and provides weak lower bounds) in an hour.

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 Section 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. In Section 3 we show how the GRLP can be transformed into the NWDSFP. Mixed-integer programming (MIP) models for both the GRLP and the NWDSFP are also proposed in Section 3; and the strength of the different models is established. 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 Section 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 were ignored and dealt with once routing paths had been determined for optical transmission, or costs associated with regenerators were ignored (Gouveia et al., 2003). Since 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 the 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 the MLSTP, these procedures are equally applicable to the MLSTP as well. Recently, 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).

Recently 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 $(1-\epsilon)\log n$. Arunabha et al. (2010) point out some errors in the Flammini et al. (2011) paper. By making the connection between the RLP and the minimum connected dominating set problem (MCDSP)[1] the authors showed 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. (2010) studied the complexity of this problem and showed that it does not admit a polynomial time approximation scheme. While the papers discussed so far focused solely on the location of

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

regenerators, recently Patel et al. (2010) considered 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 the traffic grooming problem; and discussed how to combine the two heuristics to consider both problems together.

To the best of our knowledge, neither the GRLP nor the NWDSFP has been considered previously in the literature. The variant of the NWDSFP without any node weights (i.e., the directed Steiner forest problem (DSFP)) has been studied by a few researchers in the literature. There are a handful of approximation algorithms known for the DSFP. Dodis and Khanna (1999) have shown that the DSFP cannot be approximated within $O(2^{\log^{1-\epsilon} n})$ for any fixed $\epsilon > 0$, unless NP-hard problems can be solved in quasi-polynomial time. The currently best known approximation ratio for the DSFP is $O(n^{4/5+\epsilon})$ (Feldman et al., 2009), where n is the number of nodes in the graph.

2 Preliminaries

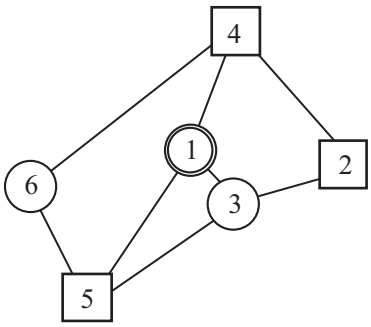
In Chen et al. (2010) we showed that the RLP is NP-complete. Since the GRLP generalizes the RLP, its NP-completeness follows immediately. Alternatively, to show the NP-completeness of the GRLP directly, one might consider a transformation from the hitting set problem (Chen et al., 2009).

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. These reduction procedures partially fix the solution by determining locations where regenerators must be placed; and use this information to reduce the size of the input graph.

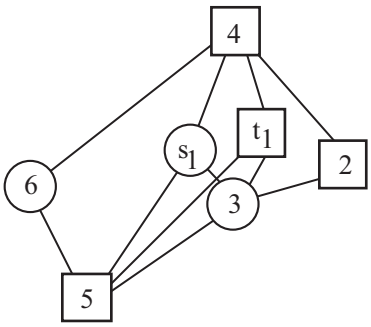
2.1 The Communication Graph

Given a graph $G = (N, F)$ with edge lengths $l : F \mapsto \mathbb{R}^+$, for each pair of nodes u and v in G, let $d(u, v)$ denote the length of the shortest path between u and v in G. We now generate a graph $B = (N, E)$, called the *communication graph*, such that $E = \{e = \{u, v\} \mid u, v \in N,\ d(u, v) \leq d_{max}\}$. In other words if 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 suffices to consider the GRLP problem on the communication graph B.

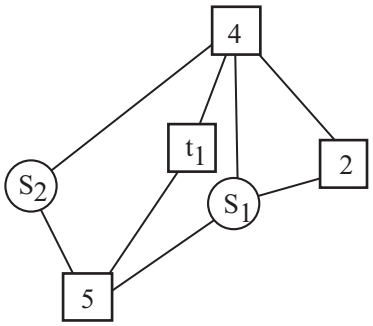
Definition 1 (GRLP on Communication Graph)**.** *Given the communication graph B, the GRLP*



(a) Graph B with $S = \{1, 3, 6\}$ and $T = \{1, 2, 4, 5\}$.



(b) Transformation into B' with $S \cap T = \emptyset$, $S' = \{s_1, 3, 6\}$ 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.

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., $S \cap T = \emptyset$, $S \cup T = N$ and $S, T \neq \emptyset$.*

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

Assume now that $S \cap T \neq \emptyset$. 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.

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

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 definition of the GRLP on the communication graph and 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 identified two S-components of B' and replaced them with *super-nodes* S_1 and S_2, as shown in Figure 1(c). The depicted instance is feasible since the two end points of every NDC node pair are connected to at least one common S-component.

In order 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 to 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:

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

(P2) If a t-node i is not in any NDC node pair, removal of 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 node l;

(P4) If the two end points of an 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) in order to reduce the input graph B. Let us denote by $L \subseteq S$, the set of locations where regenerators can be placed, and by $Q \subseteq \{1, \ldots, n_S\}$ the set of indices of S-components in which regenerators can be placed. In this iterative procedure, the variable L_{pre} denotes the set of regenerators fixed in the preprocessing procedure. When removing a node u from B, we will denote it by $B \setminus u$. 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 pseudo-code of the preprocessing procedure is given in Algorithm 1.

Our preprocessor uses a subroutine called UPDATE(B, L, Q) whose pseudo-code is provided in Algorithm 2. 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

Algorithm 1: PREPROCESSING(B, L_{pre})

$L = \emptyset$, $Q = \emptyset$, $L_{pre} = \emptyset$;
repeat
  $L_{pre} = L \cup L_{pre}$;
  UPDATE (B, L, Q);
  while $\exists v \in S$ *that satisfies (P0)* **do** $B = B \setminus \{v\}$;
  while $\exists v \in S$ *that satisfies (P1)* **do** $B = B \setminus \{v\}$;
  while $\exists u \in T$ *that satisfies (P2)* **do** $B = B \setminus \{u\}$;
  if $T = \emptyset$ **then** STOP (optimal solution found) ;
  while $\exists i \in T$ *that satisfies (P3)* **do**
    Let $l \in S$ be connected to i, l belongs to the S-component, S_j;
    $L = L \cup \{l\}$, $Q = Q \cup \{j\}$;
  while $\exists$ *NDC node pair* (i, k) *that satisfies (P4)* **do**
    Let S_j be the shared S-component;
    if $\deg_{S_j}(i) = 1$ **then**
      $L = L \cup \{l\}$, where $l \in S_j$ is the corresponding adjacent node to i;
    if $\deg_{S_j}(k) = 1$ **then**
      $L = L \cup \{w\}$, where $w \in S_j$ be the corresponding adjacent node to k;
    $Q = Q \cup \{j\}$;
until $L = \emptyset$;
return (B, L_{pre})

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 Section 4 within the construction heuristics and the post-optimization.

Figure 2 gives an example in which we obtain an optimal solution after applying the preprocessing procedure. There are three S-components in this example: $S_1 = \{s_1, s_2\}$, $S_2 = \{s_3, s_4\}$ and $S_3 = \{s_5, s_6, s_7\}$. In the first step, the preprocessor detects that all the neighbors of s_7 are connected to each other and deletes it from B. The preprocessor also detects that node t_3 has a unit S-degree. Thus, node s_4 is added to L and 2 (the index of the corresponding S-component) is added to Q. Next, the preprocessor detects that NDC node pair (t_1, t_5) shares only one S-component $S_3 = \{s_5, s_6\}$ and that t_1 is only connected to s_6 in S_3 and t_5 is only connected to s_5 in S_3. Thus, nodes s_5 and s_6 are added to L and 3 is added to Q. After that, the preprocessor detects that NDC node pair (t_2, t_4) shares only one S-component $S_2 = \{s_3, s_4\}$ and that t_2 is only connected to s_3 in S_2. Thus, node s_3 is added to L. At this point $L = \{s_3, s_4, s_5, s_6\}$ and $Q = \{2, 3\}$ and the preprocessor calls UPDATE(B, L, Q) (see Figure 2(b)). The routine adds to B edges between node pairs that can communicate after a regenerator is placed at every node in L (Figure 2(c)) and then removes from B nodes s_3, s_4, s_5, s_6, and their adjacent edges. In the resulting graph, nodes t_1, t_2, t_3, t_4, t_5 are all connected to each other (Figure 2(d)). We again repeat all preprocessing steps. Now, all t-nodes are removed from B, $L = \emptyset$, and the algorithm terminates.

Algorithm 2: UPDATE (B, L, Q)

```
for each j ∈ Q do
    Apply depth first search on S_j ∩ L and identify connected components on S_j;
Let S denote the set of all identified connected components;
for each component S_q in S do
    for each node pair (i, k) where i and k are either in S_q or connected to S_q do
        if there is no edge between i and k in B then
            add edge {i, k} to B;
Eliminate from B all nodes in L and adjacent edges;
L = Q = ∅;
return (B, L, Q)
```

We end up with an optimal solution $L_{pre} = \{s_3, s_4, s_5, s_6\}$.

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

Proof. See Appendix. □

3 Mathematical Programming Approach

In this section we first show how to transform a GRLP instance given on B into an instance of the node-weighted directed Steiner forest problem (NWDSFP) on an auxiliary graph H. Then, we study MIP formulations that are valid for the GRLP (and with a change in objective function the NWDSFP). We propose a compact and an extended formulation on graph H and a third formulation on a graph obtained by "splitting" s-nodes. For the strongest among these models (with respect to the quality of lower bounds), we propose a branch-and-cut algorithm that is described at the end of this section.

3.1 Transforming the GRLP into the 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. In order 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 therefore have either in-degree or out-degree equal to zero. The construction of H is described below. During this transformation, we will create a set of *origin-destination pairs*, denoted by D, that will correspond to the NDC node pairs from B.

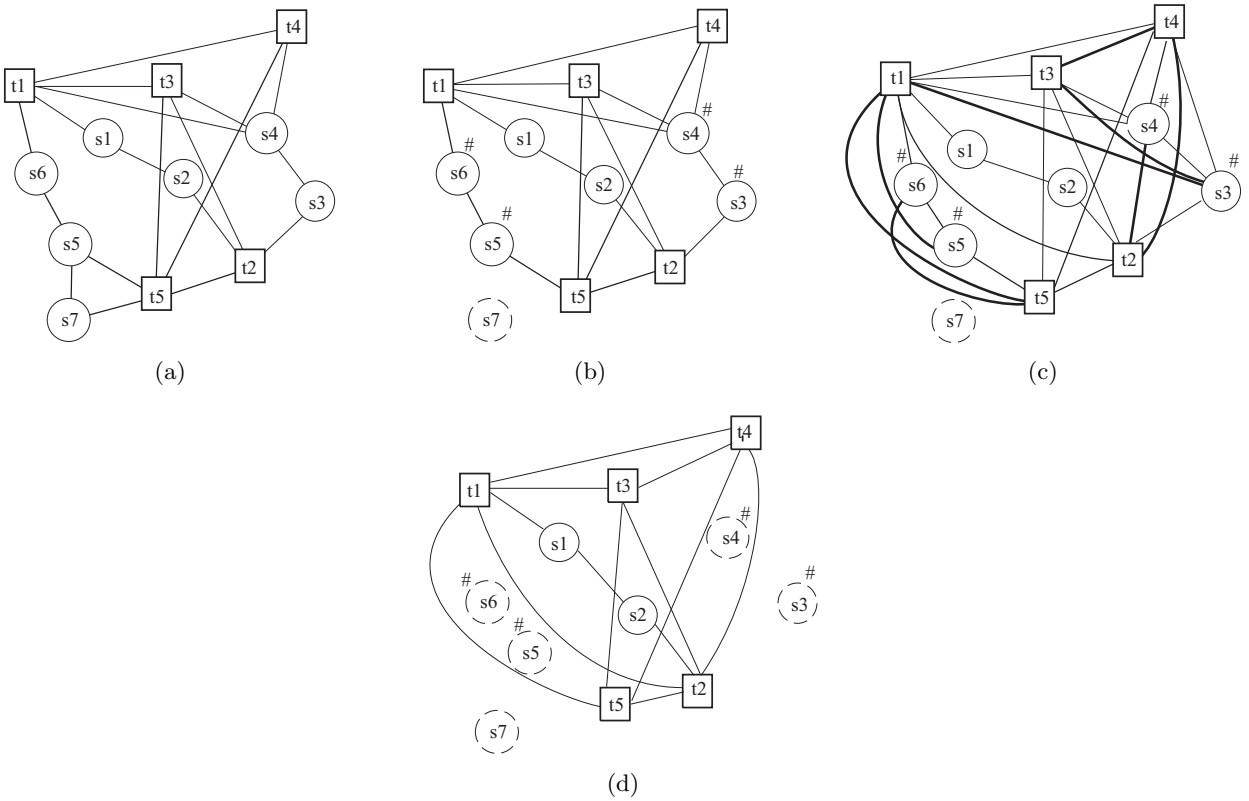


Figure 2: Illustration of the preprocessing procedure for the GRLP.

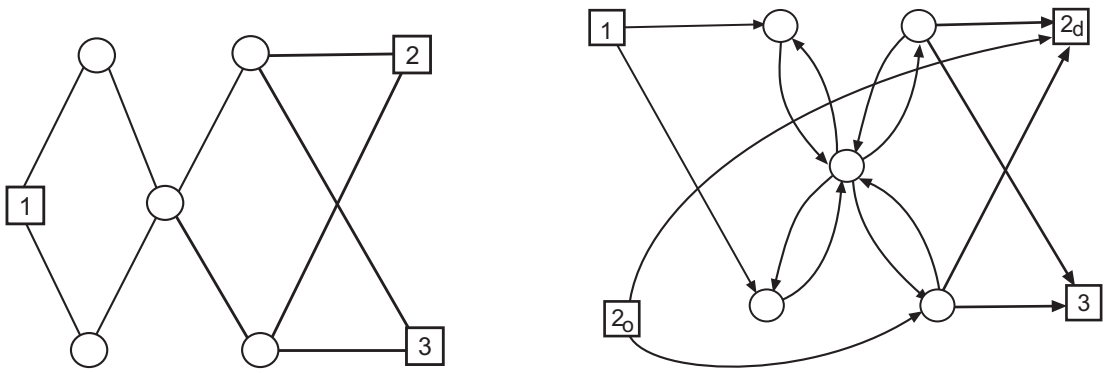
The directed graph $H = (V_H, A_H)$ is defined as follows.

$$
\begin{aligned}
V_H = V_O \cup V_D \cup S, \text{ where } V_O \text{ and } V_D \text{ are created as follows:} \\
v_o \in V_O \Leftrightarrow \exists v \in T, \exists (v, l) \in NDC \\
v_d \in V_D \Leftrightarrow \exists v \in T, \exists (l, v) \in NDC \\
A_H = \{(i,j) \mid i \in V_O, j \in S, \text{ or } i \in S, j \in V_D, \text{ or } i, j \in S\}
\end{aligned}
$$

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 3 illustrates this transformation. We now define the NWDSFP as follows.

Definition 2 (NWDSFP)**.** *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, $w : V_H \mapsto \mathrm{R}_0^+$ and $c : A_H \mapsto \mathrm{R}_0^+$, 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 $\sum_{i \in V'} w_i + \sum_{a \in A'} c_a$.*

If, in the previous definition, all node weights are equal to zero, the problem is simply referred to as the Directed Steiner Forest problem (DSFP). One can easily show that the GRLP can be



(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 3: Transformation of a GRLP instance into a NWDSFP instance.

transformed into the NWDSFP on the graph H obtained as described above.

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 NWDSFP on the graph H (whose construction is described above), and vice versa.*

Despite the term "forest" in the name of the problem, a solution to the (NW)DSFP is not necessarily a forest. In a common example in which $D = V_H \times V_H$ and all arc costs are equal to one (and all node-weights are equal to zero), the optimal solution is given by an oriented Hamiltonian cycle. Hence, in general case, an optimal solution on H is not necessarily cycle-free. However, due to the special structure of GRLP instances on H (i.e., the arc costs are zero), the following lemma shows that there always exist an optimal solution on H, whose only possible cycles are of type $(i,j) \cup (j,i)$, for some $i, j \in S$.

Lemma 4. *For any NWDSFP instance on the directed graph H constructed as above, there always exists an optimal solution $H' = (V', A')$ that induces an undirected cycle-free subgraph on S. A subgraph of H is undirected cycle-free if it is cycle-free after disregarding the arc directions.*

Proof. Assume there is an optimal solution to the NWDSFP that is not undirected cycle-free. Let C denote the cycle (disregarding directions) on H'. Since, by construction, all arcs connecting s-nodes are bidirected and with zero costs, we can eliminate the cycle C by replacing one of its arcs (i,j) by the directed path between i and j in C (adding zero cost arcs as needed) without increasing the cost of the objective function. Repeating this procedure yields the desired result. □

3.1.1 Flow and Cut Set Formulations on H

We first consider the multi-commodity flow model for the GRLP derived from the special cost structure of the corresponding NWDSFP problem on the graph $H = (V_H, A_H)$. We introduce

binary variables $y_i \in \{0,1\}, i \in V_H$ that are set to one if the node $i \in S$ is used as an intermediate node of the NWDSFP solution. Furthermore, we fix $y_i := 1$ if $i \in V_O \cup V_D$.

Every origin-destination pair $(o,d) \in D$ can now be considered as a single commodity, and we are searching for the minimal subset of nodes from S, needed to be installed in that network in order to allow origin-destination pairs to communicate with each other. In order to ensure the existence of a directed path between each origin-destination pair $k = (o,d)$, we introduce flow variables f_{ij}^k that will be set to one if the path uses the arc $(i,j) \in A_H$, and to zero otherwise. The multi-commodity flow model reads as follows.

$$(MCF_y) \qquad \min \sum_{i \in S} y_i \tag{1}$$

$$\text{s.t.} \qquad \sum_{(i,j) \in A_H} f_{ij}^k - \sum_{(j,i) \in A_H} f_{ji}^k = \begin{cases} -1, & i = d \\ 1, & i = o \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V_H,\ \forall k = (o,d) \in D \tag{2}$$

$$0 \leq f_{ij}^k \leq \min\{y_i, y_j\} \qquad \forall (i,j) \in A_H\ \forall k \in D \tag{3}$$

$$y_i = 1 \qquad \forall i \in V_O \cup V_D \tag{4}$$

$$y_i \in \{0,1\} \qquad \forall i \in S \tag{5}$$

In contrast to classical multi-commodity flow models for trees (see, e.g., Magnanti and Wolsey (1995); Vanderbeck and Wolsey (2010)), we do not use binary arc variables to define arc capacities. Since there are no arc costs involved, our capacity constraints (3) ensure that each time an arc is used, both of its end nodes are included into solution. In this and in the following models, $\mathbf{y}$ 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.

This compact model contains $O(|S|)$ binary and $O(|A_H||D|)$ continuous variables and $O(|A_H||D|)$ constraints. Unfortunately, only small instances can be solved to optimality by plugging the MCF_y model into a black-box MIP solver. Therefore, we resort to cut set based models for which efficient branch-and-cut approaches appear to be more effective in practice.

Two Equivalent Cut Set Formulations An optimal solution to the GRLP on H (and to the NWDSFP, in general) can be seen as a union of Steiner arborescences. Thereby, each node $\hat{o}$ such that $(\hat{o}, d) \in D$ defines the root of a Steiner arborescence whose terminals are all possible destination nodes having $\hat{o}$ as an origin. Therefore, the GRLP can be solved by simultaneously searching for Steiner arborescences on H with arc capacities set to $\min\{y_i, y_j\}$, for every $(i,j) \in A_H$. For each origin node $r \in V_O$, denote by $V_D(r) = \{i \in V_D \mid \exists k = (r,i) \in D\}$ the set of its destination nodes. Every subgraph of the solution that enables connection between $r \in V_O$ and all other $i \in V_D(r)$ can be seen as a Steiner arborescence rooted at r with $V_D(r)$ nodes as terminals that are leaves. The

correctness of the claim follows from the fact that, by construction, all destination nodes have only ingoing arcs, and therefore, cannot be used as interconnection nodes within the paths connecting NDC node pairs.

Given $W \subseteq V_H$, denote by $\delta^-(W) = \{(i,j) \mid i \notin W, j \in W\}$. We will also denote $\delta^-(\{i\})$ by $\delta^-(i)$. Furthermore, for a binary variable $x \in \{0,1\}^{|A_H|}$, let $x(U) = \sum_{(i,j)\in U} x_{ij}$, for all $U \subseteq A_H$. The cut set formulation that models multiple Steiner arborescences on H can be obtained from the previous flow based model by replacing inequalities (2) and (3) with the following set of exponentially many cut set inequalities

$$\sum_{(i,j)\in\delta^-(W)} \min\{y_i, y_j\} \geq 1 \quad \forall r \in V_O,\ \forall W \subseteq V_H \setminus \{r\}, W \cap V_D(r) \neq \emptyset. \tag{6}$$

The model defined by inequalities (4)-(6) will be denoted by CUT_y. Constraints (6) represent just a compact way of writing $2^{|\delta^-(W)|}$ many inequalities per each constraint of type (6). 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 $x_{ij} \in \{0,1\}$, that 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. In the corresponding model, that we will refer to as $CUT_{x,y}$, inequalities (6) are replaced by:

$$x(\delta^-(W)) \geq 1 \qquad \forall r \in V_O,\ \forall W \subseteq V_H \setminus \{r\}, W \cap V_D(r) \neq \emptyset \tag{7}$$

$$x_{ij} \leq \min\{y_i, y_j\} \quad \forall (i,j) \in A_H \tag{8}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A_H \tag{9}$$

We should note in passing that $CUT_{x,y}$ is a valid formulation for the NWDSFP as well. In the case of the NWDSFP the objective function contains both node and arc variables. Let $v_{LP}(M)$ denote the value of the LP-relaxation of an arbitrary MIP model M, and P_M denote the feasible space of its linear relaxation. The following lemma easily follows in a straightforward fashion from the max-flow min-cut theorem (Ford and Fulkerson, 1956).

Lemma 5. *The formulations CUT_y, MCF_y and $CUT_{x,y}$ are equally strong, i.e., $v_{LP}(CUT_y) = v_{LP}(MCF_y) = v_{LP}(CUT_{x,y})$. Further their projections onto the space of the* $\mathbf{y}$ *variables are identical.*

In the next section we will show that an extended formulation obtained by a disaggregation technique leads to a stronger MIP model than the ones considered above.

3.2 A Disaggregated Cut Set Formulation

In order to model each single Steiner arborescence rooted at $r \in V_O$ separately, we now disaggregate node and arc variables of the $CUT_{x,y}$ formulation. To each origin node $r \in V_O$, we assign the set of binary arc and node variables, x^r_{ij} and y^r_i, respectively, that are set to one if the arborescence rooted at r uses these arcs/nodes in order to connect to its destination nodes $V_D(r)$. The formulation reads then as follows.

$$(DCUT_{x,y}) \qquad \min \sum_{i \in S} y_i$$

$$\text{s. t.} \quad x^r(\delta^-(W)) \geq 1, \qquad \forall r \in V_O,\ \forall W \subseteq V_H \setminus \{r\}, W \cap V_D(r) \neq \emptyset \tag{10}$$

$$x^r(\delta^-(i)) = \begin{cases} y^r_i & \forall i \in S \\ 1 & \forall i \in V_D(r) \end{cases} \qquad \forall r \in V_O \tag{11}$$

$$y^r_i \leq y_i \qquad \forall r \in V_O\ \forall i \in S \tag{12}$$

$$y_i = 1 \qquad \forall i \in V_O \cup V_D \tag{13}$$

$$x^r_{ij} \in \{0,1\} \qquad \forall (i,j) \in A_H, \forall r \in V_O \tag{14}$$

$$y^r_i, y_i \in \{0,1\} \qquad \forall i \in V_H, \forall r \in V_O \tag{15}$$

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.

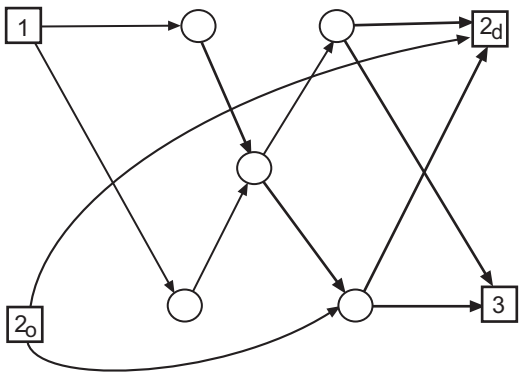
Lemma 6. *The model $DCUT_{x,y}$ is strictly stronger than the model $CUT_{x,y}$, i.e., $v_{LP}(DCUT_{x,y}) \geq v_{LP}(CUT_y)$, and there exist GRLP instances for which the strict inequality holds.*

Proof. To prove this result it suffices to consider minimal feasible solutions $(\mathbf{x'^r}, \mathbf{y'^r}, \mathbf{y'}) \in P_{DCUT_{x,y}}$. Let $x'_{ij} = \max_r x'^r_{ij}$. Then it is easy to verify that $(\mathbf{x'}, \mathbf{y'})$ is also feasible to inequalities (4) and (7). With regards to inequality (8), 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 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 4 illustrates an example that shows that $v_{LP}(DCUT_{x,y})$ can be strictly greater than $v_{LP}(CUT_y)$. □

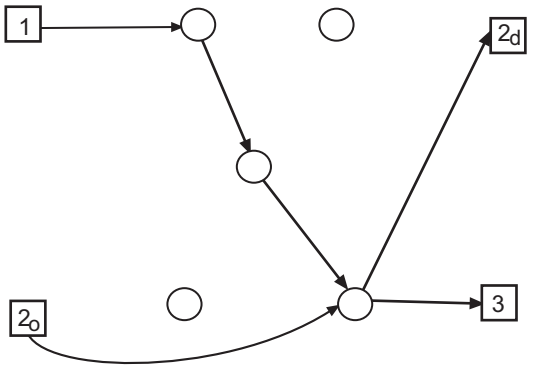
While the above lemma is stated in the context of the GRLP, it should be clear that $DCUT_{x,y}$ is a valid formulation for the NWDSFP and Lemma 6 also holds for the NWDSFP.

3.3 Node Splitting Transformation of the GRLP

Node splitting has been shown to be a very useful technique for different network design problems (see e.g., Ahuja et al., 1993; Chen et al., 2010; Gamvros et al., 2012). We now use the node splitting to develop a strong MIP formulation on an extended graph. The new model does not use



(a) An optimal LP-solution of the model $CUT_{x,y}$. For all depicted (i,j), $x_{ij} = 0.5$, and for all $i \in S$, $y_i = 0.5$. The optimal objective value $v_{LP}(CUT_{x,y}) = 2.5$. This solution is infeasible for the model $DCUT_{x,y}$ because the s-node in the middle violates constraints (12).



(b) An optimal LP-solution of the model $DCUT_{x,y}$. $v_{LP}(DCUT_{x,y}) = 3$. For all depicted (i,j), $x_{ij} = 1$, and for all non-isolated $i \in S$, $y_i = 1$.

Figure 4: Example for proof of Lemma 6

disaggregation and therefore it has significantly fewer variables than the model $DCUT_{x,y}$, while keeping the same quality of lower bounds.

The graph H is transformed into an *extended directed graph* $\tilde{H} = (\tilde{V}_H, \tilde{A}_H)$ such that:

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

In other words, for every node i in S we create two nodes i' and i'' corresponding to 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. Figure 5 illustrates this transformation and it is easy to see that the following result holds:

Lemma 7. *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 $\tilde{H}$ (whose construction is described above), and vice versa.*

Directed Cut Set Formulation on $\tilde{H}$ For every arc $(i,j) \in \tilde{A}_H$, we introduce a binary variable $z_{ij} \in \{0,1\}$ that is set to one 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 to zero, otherwise. The following cut set formulation models multiple Steiner arborescences on $\tilde{H}$, and is valid for both the GRLP and the DSFP (as well as the NWDSFP

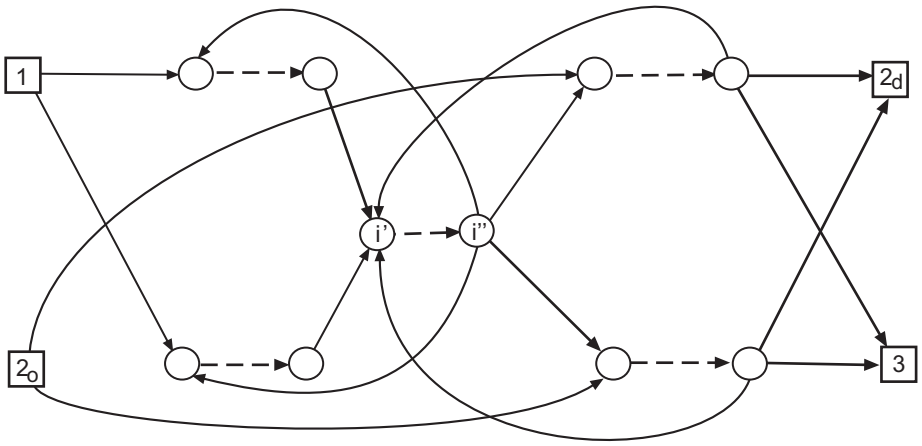


Figure 5: Node-splitting transformation of the graph given in Figure 3(b). Dashed arrows correspond to the s-nodes that are split.

after transformation to an extended directed graph where the nodes are split).

$$\min\{\sum_{(i,j)\in\tilde{A}_H} c_{ij}z_{ij} \mid \sum_{(i,j)\in\delta^-(W)} z_{ij} \geq 1,\ \forall r \in V_O,\ W \subseteq \tilde{V}_H \setminus \{r\}, W \cap V_D(r) \neq \emptyset, z_{ij} \in \{0,1\}^{|\tilde{A}_H|}\}$$

We will refer to this model as $SCUT$.

Theorem 1. *Formulations $SCUT$ and $DCUT_{x,y}$ are equally strong, i.e., $v_{LP}(SCUT) = v_{LP}(DCUT_{x,y})$.*

Proof. $v_{LP}(SCUT) \geq v_{LP}(DCUT_{x,y})$: To prove this inequality, it suffices to consider minimal feasible solutions from P_{SCUT}. Given a minimal feasible $\hat{\mathbf{z}} \in P_{SCUT}$, we will show that there always exists a solution $(\hat{\mathbf{x}}^{\mathbf{r}}, \hat{\mathbf{y}}^{\mathbf{r}}, \hat{\mathbf{y}})$ feasible for the $DCUT_{x,y}$ 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 multi-commodity 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 $\hat{\mathbf{x}}^{\mathbf{r}}$ and $\hat{\mathbf{y}}^{\mathbf{r}}$ as follows.

$$\hat{x}_{ij}^r = \begin{cases} \max_{d\in V_D(r)} f_{i''j'}^{(r,d)}, & i,j \in S \\ \max_{d\in V_D(r)} f_{i,j'}^{(r,d)}, & i \in V_O, j \in S \\ \max_{d\in V_D(r)} f_{i'',j}^{(r,d)}, & i \in S, j \in V_D \end{cases} \quad r \in V_O,\ (i,j) \in A_H \quad \text{and} \quad \hat{y}_i^r = \hat{x}^r(\delta^-(i)),\ i \in S$$

By construction of the flow and the definition of $\hat{\mathbf{x}}^{\mathbf{r}}$ variables, 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 $\hat{\mathbf{y}}^{\mathbf{r}}$ variables, constraints (10) and (11) are satisfied. The constructed vector $(\hat{\mathbf{x}}^{\mathbf{r}}, \hat{\mathbf{y}}^{\mathbf{r}}, \hat{\mathbf{y}})$ also satisfies constraints (12), 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 $\hat{y}_i^r \leq \hat{y}_i$ for some $r \in V_O$ and $i \in S$. This implies that, without loss of generality, there exist 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, re-routing the flow of one of the two commodities still induces a feasible solution, with at least one of $\hat{\mathbf{z}}$ capacities being reduced. However, this is in contradiction with our assumption that $\hat{\mathbf{z}}$ is a minimal feasible solution in P_{SCUT}.

$v_{LP}(DCUT_{x,y}) \geq v_{LP}(SCUT)$: Given any feasible solution $(\hat{\mathbf{x}}^{\mathbf{r}}, \hat{\mathbf{y}}^{\mathbf{r}}, \hat{\mathbf{y}}) \in DCUT_{x,y}$, we construct a solution $\hat{\mathbf{z}} \in P_{SCUT}$ by setting $\hat{z}_{i'i''} = \hat{y}_i$, for all $i \in S$, and $\hat{z}_{ij} = 1$, 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 $SCUT$.

□

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 to look at the following family of cut sets

$$\mathcal{W} = \bigcup_{r\in V_O} \{W \mid W \subseteq \tilde{V}_H \setminus \{r\}, W \cap V_D(r) \neq \emptyset \text{ and } \delta^-(W) \cap \tilde{A} = \emptyset\}. \tag{16}$$

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 other words we can rewrite the above model for the GRLP in terms of the "split arcs" as follows.

$$(NSCUT) \qquad \min \sum_{(i',i'')\in\tilde{A}_S} z_{i'i''}$$

$$\text{s. t.} \sum_{(i',i'')\in\delta^-(W)} z_{i'i''} \geq 1, \qquad \forall W \in \mathcal{W} \tag{17}$$

$$z_{i'i''} \in \{0,1\} \quad \forall i \in S \tag{18}$$

Notice that, after projecting back the arc variables $z_{i'i''}$ into the node variables y_i, for all $i \in S$, the cut sets defined by (16) correspond to the (r,d) *separating-node-cuts* 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 from H results in a graph in which the nodes r and d are disconnected. Therefore, it suffices to consider minimal separating-node-cuts to define the model $NSCUT$.

3.4 Branch-and-Cut Algorithm

For solving the GRLP to optimality, among all models presented above, we chose the $NSCUT$ model. As discussed previously, this model exhibits the best lower bounds and involves significantly

Algorithm 3: OrderingNDCPairs(NDC)

```
D = ∅;
repeat
    Let r ∈ 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 ∪ {r}, V_D = V_D ∪ {i}, D = D ∪ {(r, i)};
        NDC = NDC \{(r, i), (i, r)};
until NDC is empty;
return D
```

fewer variables than the equally strong disaggregated model $DCUT_{x,y}$. The main algorithmic issues of an efficient implementation of the branch-and-cut framework for solving the $NSCUT$ formulation are provided below.

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 3.

3.4.2 Initialization

Recall that in the enhanced directed graph $\tilde{H}$, each s-node $i \in S$, is split into arc $(i', i'') \in \tilde{A}_S$. 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: $\sum_{i'i'':i''k \in \tilde{A}} z_{i'i''} \geq 1$, 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 need to be passed through: $\sum_{i'i'':ri' \in \tilde{A}} z_{i'i''} \geq 1$, for all $r \in V_O$. Both groups of inequalities are special cases of cut set inequalities (17).

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. Similarily, 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: $z_{i'i''} \leq \sum_{k'k'':i''k' \in \tilde{A}} z_{k'k''}$ and $z_{i'i''} \leq \sum_{k'k'':k''i' \in \tilde{A}} z_{k'k''}$, for all $i \in S^{in}$.

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 Section 6, we did not add any cuts in the initialization phase except the trivial ones ($0 \leq z_{i'i''} \leq 1$, for all $i \in S$).

3.4.3 Separation

The cut set inequalities (17) can be separated in polynomial time. The separation algorithm relies on the calculation of multiple maximum flows. Here, we explain an efficient way to do that. Given a fractional solution $\mathbf{z}'$ of the current LP-relaxation, we define the capacities on the arcs of the *support graph* as follows:

$$cap_{uv} = \begin{cases} z'_{i'i''}, & (u,v) = (i', i''), i \in S \\ 1, & \text{otherwise} \end{cases}$$

This way, only the cut sets from the collection $\mathcal{W}$ defined above, i.e., cut sets involving split node variables only, 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 cut sets is built. Using nested and back-cuts (as described in e.g., Ljubić and Gollowitzer (2012)) up to 100 cuts are added into this pool, and then inserted into the master LP whose value is then recalculated. Detected violated cut sets 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 (2012)), i.e., during the separation procedure, among all cut sets with equal violation, those with the smallest cardinality are preferred.

4 Heuristic Approach

As the exact approach presented above 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 works then 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. The pseudo-code of the procedure PLACE_REGENS that identifies L_j for a given component S_j and node t is provided in Algorithm 4. We illustrate the algorithm in two examples given below. 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.

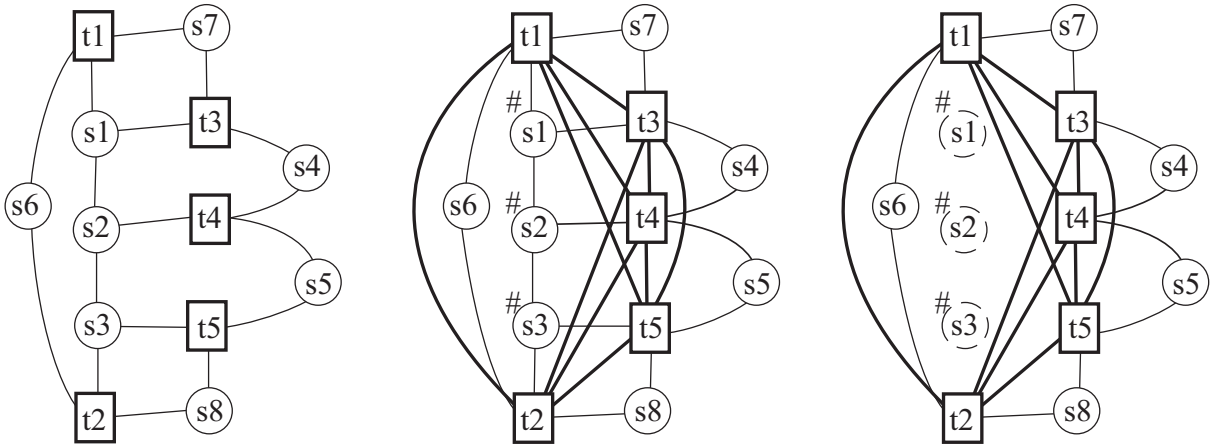


Figure 6: Illustration of Heuristic GH1: Input graph B and a solution found by GH1.

GH1 deploys a regenerator at every node in L_t^{max}. We then update the graph B (using Algorithm 2 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 6 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, $\frac{10}{3}$ 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 particular case, after we updated the graph, we already 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 select an arbitrary s-node $s^* \in S_j$ adjacent to t and perform a *restricted breath-first-search* (BFS) (denoted by RESTRICTED_BFS in Algorithm 4) in which the BFS search 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 $T_{BFS} = (V_{BFS}, E_{BFS})$. 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 and 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 the breath-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

Algorithm 4: PLACE_REGENS(S_j, t)

```
Arbitrary select a neighboring s-node of t, say s* ∈ S_j;
T_BFS = (V_BFS, E_BFS) = RESTRICTED_BFS(S_j ∪ T, s*);
Mark all s-nodes from V_BFS ∩ S_j as "unmerged";
Iteratively eliminate from T_BFS s-nodes that are leaves, until there are no such nodes left;
while ∃s ∈ S_j ∩ V_BFS labeled as "unmerged" do
    Select a longest (from s*) "unmerged" branch B in T_BFS;
    for each s-node l ∈ B do
        for each e = {k, l} ∈ E do
            if k ∈ V_BFS and e ∉ E_BFS then
                Delete the edge from k to its predecessor in E_BFS;
                E_BFS = E_BFS ∪ {e};
        Mark l as "merged";
    Iteratively eliminate from V_BFS s-nodes that are leaves, until there are no such nodes left;
return L_j = V_BFS ∩ S_j
```

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}.

Figure 7 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 $s^* = s_1$. T_{BFS} is constructed through Figure 7(a)-(c). The merging process is shown in Figure 7(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 by $\{s_7, t_2\}$ 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 $L_1 = \{s_1, s_4, s_6, s_7\}$. 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 $w(L_1) = \frac{6}{4} = 1.5$.

Lemma 8. *Heuristic GH1 runs in $O(|T||S||N|^2)$ time.*

Proof. See Appendix. □

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 how we place regenerators, once the BFS tree T_{BFS} is found. As opposed to GH1, GH2 focus on placing regenerators on only 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.

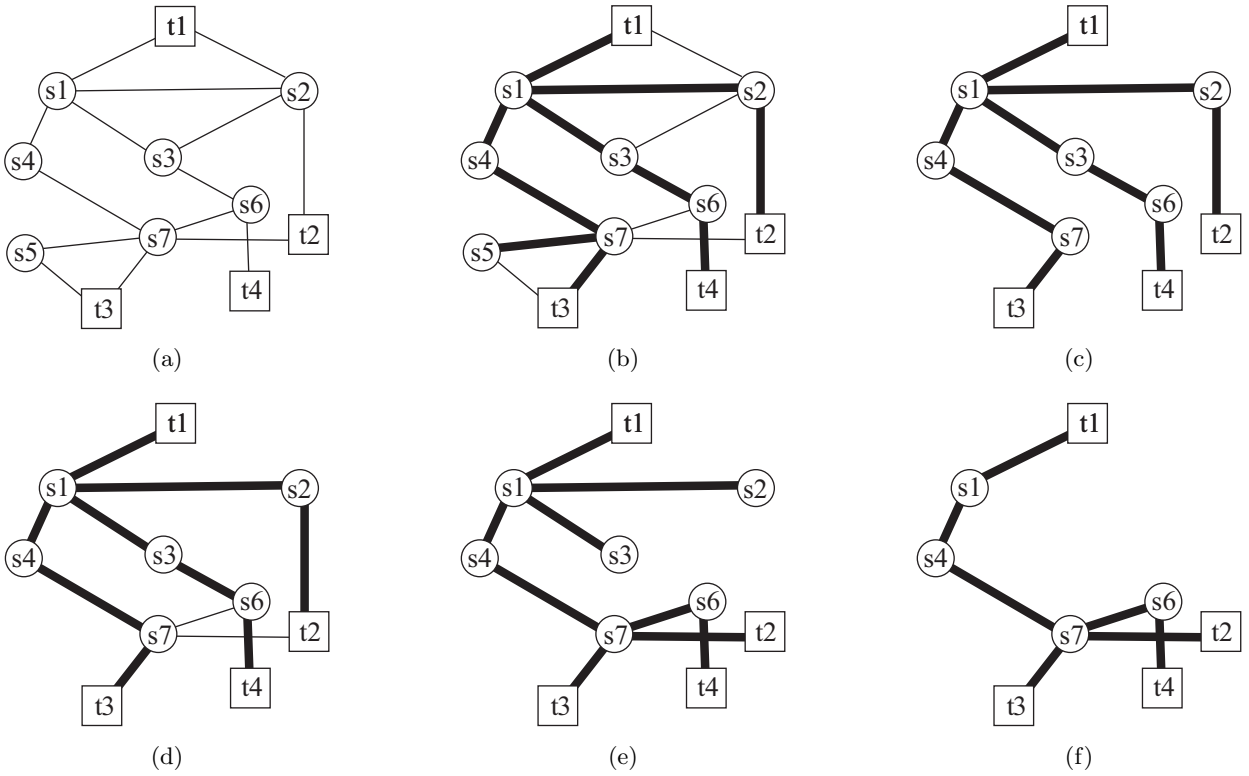


Figure 7: Illustration of the RESTRICTED_BFS ((a)-(c)) and PLACE_REGENS ((d)-(f)) procedure.

The graph B is then updated (using the UPDATE-procedure given in Algorithm 2), a new t-node with the lowest degree in 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 procedure (POLS) consists of a basic subroutine called **p-for-q**. POLS is applied to the solutions obtained from GH1 or GH2 in order to improve upon them. **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 $Q \subseteq S_i \setminus L_i$. 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 $Q \subseteq S_j \setminus L_j$.

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

Proof. See Appendix. □

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 problem (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 GRLP is to minimize $\sum_{i \in L} 1 = |L|$.

The construction of the communication graph B and the preprocessing procedure described in Section 2 do not depend on weights and thus remain unchanged for the WGRLP. Regarding the MIP formulations proposed in Section 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 in 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 11.0) and Concert Technology (version 2.9) (CPLEX, 2012). Our experiments were performed on an Intel Core 2 Duo 3 GHz machine with 3.25 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 of randomly generated networks 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 below guarantees that the sets S and T are disjoint and that the instances are feasible.

Set1 (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 t-nodes among them. The number of S-components, n_S, is a random integer value chosen from $\{2,\ldots,5\}$ with equal probability. Each s-node is randomly assigned to one of the S-components with probability $\frac{1}{n_S}$. Notice that the probability of a particular S-component being empty is $\Pi_{i=1}^{|S|}(\frac{n_S-1}{n_S})$. 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 $d_i = \frac{2|E_i|}{|S_i|(|S_i|-1)}$, 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 fifty-fifty chance of getting assigned either a high density ($d_i = 70\%$) or a 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 $\{0, \frac{1}{|S_j|}, \frac{2}{|S_j|}, ..., \frac{|S_j|-1}{|S_j|}\}$ 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 (not connected to any s-node) is $\Pi_{j=1}^{n_S}(\frac{1}{|S_j|})$. When this happens, we arbitrarily choose a component S_i and randomly select $q \in \{\frac{1}{|S_i|}, \frac{2}{|S_i|}, ..., \frac{|S_i|-1}{|S_i|}\}$ 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 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 $p \in \{0.25, 0.5, 0.75\}$.

Set2 (WGRLP Instances): We generate instances for weighted GRLP 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.

Set3 (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 Section 3, and by running the two heuristic frameworks described in Section 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 "$p(\%)$". 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, "#Opt" reports the number of optimal solutions obtained by the heuristic framework (in case 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. "#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. "#Opt" column shows in how many (out of 10) instances, B&C solved the the underlying instance to optimality. For the set of WGRLP instances (Set2), 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.

Set1. Out of 150 instances of this group, B&C solves all but two 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 t-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. Lowering the values

of p (i.e., setting $p = 0.25$) implies less NDC node pairs, and therefore, less cutting planes that need to be inserted before the optimal solution is found. On the other hand, increasing the values of p (i.e., setting $p = 0.75$) implies lower percentage of s-nodes, and therefore, the reduction of the size of the search space (as there are fewer $\mathbf{z}$ 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 two remaining cases), we observed that in 134 cases optimal solutions were found, for 12 instances, this difference was equal to one, and for the remaining four cases, this difference was equal to two. Both heuristic approaches solved each single Set1 instance within a few seconds, whereas the running times of the branch-and-cut algorithm exponentially increase with the number of nodes and the number of NDC node pairs.

Comparing between the two heuristic frameworks we find that the GH1-Framework provides the best solution in 146 instances, finds the optimal solution in 128 instances, and the POLS was useful (i.e., found an improvement to the greedy solution) in 143 instances. The GH2-Framework provides the best solution in 143 instances, finds the optimal solution in 125 instances, and the POLS was useful in 141 instances. The GH1-Framework solely provides the best heuristic solution in 7 instances, while the GH2-Framework solely provides the best heuristic solution in 4 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 GH2, indicating that the values of GH2 were slightly worse than those obtained by GH1. We may also conclude that the POLS plays a significant role in the performance of the proposed heuristic frameworks.

Set2. Table 2 reports computational results for Set2. The B&C procedure solves all 150 instances to optimality, and needs on average less time than for solving the same instances of Set1. This can be explained by the fact that weighted GRLP solutions are less symmetric than the corresponding unweighted ones. The better of the two heuristic frameworks finds optimal solution in 134 cases, in 10 cases the difference between the heuristic and the optimal solution is equal to one, and in the remaining 6 cases this difference varies between 2 and 5. Comparing between the two heuristic frameworks we find that the GH1-Framework provides the best solution in 134 instances, finds the optimal solution in 120 instances, and the POLS was useful in all but one instances. The GH2-Framework provides the best solution in 143 instances, finds the optimal solution in 131 instances, and the POLS was useful in all instances. In 8 instances the GH1-Framework was solely the best heuristic approach, while 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 Set1, the running times of the two frameworks are about the

		GH1-Framework					GH2-Framework					B&C			
n	$p(\%)$	NR	RT	#Opt	#PO	#Imp	NR	RT	#Opt	#PO	#Imp	UB	LB	RT	#Opt
50	25	7.2	0.0	10	1.5	9	7.2	0.0	10	1.9	9	7.2	7.2	0.34	10
	50	7.7	0.0	8	1.6	7	7.7	0.0	8	2	7	7.5	7.5	0.50	10
	75	4.3	0.0	10	2.1	7	4.3	0.0	10	2.7	8	7.5	7.5	0.50	10
75	25	9.8	0.0	9	1.2	9	9.8	0.0	9	1.8	10	9.7	9.7	2.21	10
	50	9.5	0.0	8	3.8	10	9.6	0.0	7	5	10	9.2	9.2	5.67	10
	75	5.6	0.0	8	2.9	9	5.5	0.0	9	3.6	9	5.5	5.5	4.03	10
100	25	11.8	0.0	9	1.7	10	11.8	0.0	9	3.1	10	11.7	11.7	11.30	10
	50	11.0	0.0	8	3.9	10	10.9	0.0	9	6.5	10	10.8	10.8	38.64	10
	75	7.3	0.0	6	3.1	10	7.2	0.2	7	4.3	10	7.0	7.0	16.59	10
125	25	14.1	0.0	10	3.1	10	14.1	0.0	10	3.9	10	14.1	14.1	45.77	10
	50	12.0	0.4	9	5.1	10	12.1	0.3	9	6.5	10	11.9	11.9	290.56	10
	75	8.5	0.1	6	5.4	10	8.4	0.1	7	5.9	10	8.1	8.1	110.69	10
150	25	14.5	0.1	10	4.1	10	14.5	0.5	10	4.4	10	14.5	14.5	220.49	10
	50	12.3	1.3	8	6.9	10	12.3	0.7	8	9.4	10	12.2	12.0	1188.84	9
	75	9.5	0.3	6	4.9	10	9.5	0.1	6	6.5	10	9.1	8.9	903.26	9

Table 1: Computational results for Set1

		GH1-Framework					GH2-Framework					B&C			
n	$p(\%)$	Obj	RT	#Opt	#PO	#Imp	Obj	RT	#Opt	#PO	#Imp	UB	LB	RT	#Opt
50	25	21.4	0.0	9	5.5	10	21.4	0.0	9	6.8	10	21.3	21.3	0.33	10
	50	21.3	0.0	10	7.1	10	21.4	0.0	9	9.2	10	21.3	21.3	0.52	10
	75	11.5	0.0	8	8.3	9	11.5	0.0	8	11.5	10	11.3	11.3	0.21	10
75	25	28.4	0.0	9	5.4	10	28.4	0.0	9	6.6	10	28.2	28.2	2.00	10
	50	25.8	0.0	9	15.9	10	26.1	0.0	8	16.5	10	25.7	25.7	4.52	10
	75	14.2	0.0	8	9.8	10	14.5	0.0	8	12.8	10	13.9	13.9	2.28	10
100	25	35.0	0.0	10	6.9	10	35.1	0.0	9	10.9	10	35.0	35.0	9.73	10
	50	28.9	0.2	9	16	10	28.8	0.1	9	24.1	10	28.6	28.6	26.81	10
	75	18.9	0.3	8	14.3	10	19.1	0.1	6	14.6	10	18.7	18.7	12.54	10
125	25	42.6	0.1	10	13.3	10	42.6	0.4	10	16.7	10	42.6	42.6	41.11	10
	50	32.3	0.6	8	18.1	10	32.5	0.9	7	24.8	10	31.6	31.6	181.41	10
	75	20.9	0.2	7	20.2	10	20.8	0.1	5	22.4	10	20.1	20.1	58.80	10
150	25	42.1	0.3	9	15.5	10	42	0.3	9	19.4	10	41.9	41.9	159.69	10
	50	32.8	1.8	8	26.3	10	33.3	1.4	7	33.3	10	32.3	32.3	728.19	10
	75	21.9	0.5	9	21.8	10	22.1	0.8	7	24.3	10	21.4	21.4	199.15	10

Table 2: Computational results for weighted instances (Set2)

same.

Table 3 compares the number of branch-and-bound nodes (BBnodes) in the B&C procedure on Set1 and Set2. BBnodes = 0 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. Further, the number of BBnodes significantly increases as p gets closer to 50%.

Set3. Table 4 shows the corresponding results for the large-scale instances (Set3). Since the running time of the POLS increases with the number of nodes, for graphs with $n \geq 300$ we

		Set1			Set2		
n	$p(\%)$	Median	Mean	BBnodes = 0	Median	Mean	BBnodes = 0
50	25	1.00	1.20	5	0.00	0.90	6
50	50	2.00	2.80	2	2.00	2.10	4
50	75	3.00	2.50	3	0.00	1.30	7
75	25	2.00	2.60	4	2.00	2.50	3
75	50	4.00	23.70	0	5.00	13.70	0
75	75	10.50	11.60	0	7.00	9.20	0
100	25	4.50	11.10	1	5.00	6.30	1
100	50	38.50	57.40	0	19.00	44.70	0
100	75	16.50	22.20	0	9.00	14.60	0
125	25	21.50	23.50	0	18.50	26.20	0
125	50	200.00	677.80	0	188.50	192.40	0
125	75	48.50	83.70	0	29.00	43.60	0
150	25	34.00	225.60	0	48.50	81.00	0
150	50	381.50	561.60	0	151.50	428.50	0
150	75	178.50	212.20	0	43.00	72.00	0

Table 3: Number of Branch and Bound nodes (Set1 and Set2).

used only **2-for-1** local search. Out of 150 instances, the branch-and-cut procedure finds optimal solutions only in 28 cases. Out of these 28 instances, the heuristics find the optimal solution in 23 cases. Comparing between the two heuristic approaches we find that the GH1-Framework provides the best solution in 127 instances, finds the optimal solution in 23 instances, and the POLS was useful in all instances. The GH2-Framework provides the best solution in 115 instances, finds the optimal solution in 21 instances, and the POLS was useful in all instances. In 35 instances the GH1-Framework was solely the best heuristic approach, while the GH2-Framework was solely the best heuristic approach in 23 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 that 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 modelling location-dependent regenerator costs. Several safe preprocessing procedures are introduced. Application of these procedures may already lead to an optimal solution, if the underlying instance is sufficiently small / simple.

There are two results that are valid for the RLP (a special case of GRLP, which we have studied previously in Chen et al. (2010)), but that cannot be directly applied to the GRLP: (1) the RLP is equivalent to the maximum leaf spanning tree problem, and (2) the RLP problem can be modeled

		GH1-Framework					GH2-Framework					B&C			
n	$p(\%)$	NR	RT	#Opt	#PO	#Imp	NR	RT	#Opt	#PO	#Imp	UB	LB	RT	#Opt
175	25	16.5	0.3	9	4.8	10	16.4	0.3	9	7.3	10	16.3	16.3	1159.39	10
	50	13.4	2.2	1	7.8	10	13.2	2.0	2	8.7	10	13.6	11.2	3218.76	2
	75	9.6	0.7	3	7	10	9.7	0.6	3	7.7	10	9.3	8.6	1838.60	7
200	25	17.5	0.6	3	4.4	10	17.4	0.9	4	7.5	10	17.5	15.8	2652.17	4
	50	13.0	6.4	0	8.1	10	13.0	5.9	0	9.9	10	14.4	8.0	3601.09	0
	75	10.5	3.6	4	7.6	10	10.6	3.6	4	9.9	10	10.4	9.0	3004.27	5
300	25	17.4	1.4	0	5.5	10	16.8	1.7	0	7.9	10	–	10.5	3600	0
	50	16.1	2.6	0	6.5	10	15.4	2.0	0	9.4	10	–	7.4	3600	0
	75	12.9	2.1	0	6.7	10	12.5	2.1	0	8.4	10	–	7.8	3600	0
400	25	22.4	9.2	0	7	10	22.4	8.2	0	9.6	10	–	7.3	3600	0
	50	20.0	20.0	0	10.4	10	19.8	24.3	0	11.4	10	–	5.9	3600	0
	75	14.4	6.0	0	10.2	10	14.7	5.9	0	11.6	10	–	7.0	3600	0
500	25	24.8	29.8	0	9.5	10	25.0	35.3	0	11.3	10	–	5.4	3600	0
	50	21.4	34.8	0	11.7	10	21.4	34.7	0	13.5	10	–	4.8	3600	0
	75	15.0	32.0	0	9	10	12.7	37.0	0	8.4	10	–	5.2	3600	0

Table 4: Computational results for large scale instances (Set3)

as a Steiner arborescence problem on directed graphs. Contrary to these results, the GRLP is a new combinatorial optimization problem on undirected graphs, not known in the literature yet. When transformed into a directed graph, we show that the GRLP can be transformed into a (node-weighted) directed Steiner forest problem. For the latter case, we derived several mixed integer linear programming formulations. We provided a theoretical comparison of the quality of lower bounds obtained by solving LP-relaxations of the corresponding models. 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 (Set1) and 150 weighted (Set2) instances with between 50 and 150 nodes and with different densities. Additionally, we tested 150 larger graphs (Set3) 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 124 out 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.

Acknowledgements

This work has been partially done during the research stay of I. Ljubić at the Decision, Operations, and Information Technologies Department, The Robert H. Smith School of Business, University of Maryland. I. Ljubić is supported by the APART Fellowship of the Austrian Academy of Sciences (OEAW). This support is greatly appreciated.

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows*. Prentice Hall.

Arunabha, A. Sen, S. Banerjee, P. Ghosh, S. Murthy, H. Ngo. 2010. Brief announcement: on regenerator placement problems in optical networks. *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*. SPAA '10, ACM, New York, NY, USA, 178–180.

Borella, M.S., J.P. Jue, D. Banerjee, B. Ramamurthy, B. Mukherjee. 1997. Optical components for WDM lightwave networks. *Proc IEEE*, vol. 85. 1274–1307.

Chen, S., I. Ljubić, S. Raghavan. 2009. The generalized regenerator location problem. *Proceedings of the International Network Optimization Conference (INOC 2009)*.

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

Chen, S., S. Raghavan. 2007. The regenerator location problem. *Proceedings of the International Network Optimization Conference (INOC 2007)*.

Cherkassky, B. V., A. V. Goldberg. 1997. On implementing push-relabel method for the maximum flow problem. *Algorithmica* **19** 390–410.

CPLEX, IBM. 2012. `http://www.ilog.com/products/cplex/`. Last seen on March 10th 2012.

Dodis, Y., S. Khanna. 1999. Designing networks with bounded pairwise distance. *Annual ACM Symposium on the Theory of Computing*. ACM, 750–759.

Feldman, M., G. Kortsarz, Z. Nutov. 2009. Improved approximating algorithms for directed Steiner forest. *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 922–931.

Fernandes, M. L., L. Gouveia. 1998. Minimal spanning trees with a constraint on the number of leaves. *European Journal of Operational Research* **104**(1) 250–261.

Flammini, M., A. Marchetti-Spaccamela, G. Monaco, L. Moscardelli, S. Zaks. 2011. On the complexity of the regenerator placement problem in optical networks. *IEEE/ACM Trans. Netw.* **19**(2) 498–511.

Ford, Jr., L. R., D. R. Fulkerson. 1956. Maximal flow through a network. *Canadian Journal of Mathematics* **8** 399–404.

Gamvros, I., L. Gouveia, S. Raghavan. 2012. Reload cost trees and network design. *Networks* doi: 10.1002/net.20443. To appear.

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

Ljubić, I., S. Gollowitzer. 2012. Layered graph approaches to the hop constrained connected facility location problem. *INFORMS Journal on Computing* To appear.

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

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

Magnanti, T., L. Wolsey. 1995. Optimal trees. *Handbook in Operations Research and Management Science* 503–615.

Mertzios, G., I. Sau, M. Shalom, S. Zaks. 2010. Placing regenerators in optical networks to satisfy multiple sets of requests. S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, P. G. Spirakis, eds., *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP), Bordeaux, France*. 333–344.

Mukherjee, B. 2000. WDM optical communication networks: progress and challenges. *IEEE J Selected Areas Commun* **18** 1810–1824.

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

Patel, A., C. Gao, J. P. Jue, X. Wang, Q. Zhang, P. Palacharla, T.Naito. 2010. Traffic grooming and regenerator placement in impairment-aware optical WDM networks. *14th Conference on Optical Network Design and Modeling (ONDM)*. 1–6.

Rumley, S., C. Gaumier. 2009. Cost aware design of translucent WDM transport networks. *Proceedings of the 11th International Conference on Transparent Optical Networks, Azores, Portugal*.

Vanderbeck, F., L. A. Wolsey. 2010. Reformulation and decomposition of integer programs. M. Jünger, G. L. Nemhauser, W. R. Pulleyblank, D. Naddef, T. M. Liebling, eds., *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer, 431–504.

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

Zymolka, A. 1999. Untersuchung eines Tourenplanungsproblems. Master thesis, Mathematics, Philipps-University Marburg. Germany.

Appendix

Proof of Lemma 2: To check if all the neighbors of an s-node are fully connected the preprocessor needs to compare the adjacency list of this node with the adjacency lists of each of its neighbors. Thus, testing the property (P1) takes $O(|S||E|)$ time. It takes $O(|T|)$ time to scan for t-nodes with either degree 1 or $|T|-1$ (properties (P2) and (P3)). For each NDC node pair (i,k) it takes $O(n_S)$ time to check if it meets the first criterion of (P4) and $O(\deg_S(i)+\deg_S(k))$ time to check if it meets the second criterion of (P4). Both n_S and $\deg_S(.)$ are bounded by $|S|$. Thus, property (P4) can be checked in $O(|S|)$ time for each NDC node pair. As the number of NDC node pairs is bounded by $|T|^2$, the entire check takes $O(|T|^2|S|)$ time. Regarding the complexity of the UPDATE procedure, it takes $O(|E_{S_j}|)$ time to conduct the depth-first-search on each S_j and identify $\mathcal{S}$. Next, for each S_q in $\mathcal{S}$ it takes $O(|E|)$ time to identify all the nodes connected to S_q (the adjacency list of every node in S_q needs to be scanned) and subsequently $O(|N|^2)$ to identify the edges that need to be added to B. The size of $\mathcal{S}$ is bounded by $|S_j|$. Thus, the preprocessor spends $O(|S_j|(|N|^2+|E|)+|E_{S_j}|)=O(|S_j||N|^2)$ on each S_j. The entire UPDATE procedure takes $O(\sum_{j\in Q}(|S_j||N|^2)=O(|S||N|^2)$ time. The outer while loop is repeated for at most $|S|$ times. Therefore the entire preprocessing procedure takes $O(|S|^2|N|^2)$ time in the worst case.

□

Proof of Lemma 8: The procedure is dominated by: (1) the BFS step which takes $O(|E_j| + |S_j||T|)$ time, and (2) the merging step which takes $O(|S_j||S_j + T|)$ time. Thus it takes $O(\sum_{j=1}^{n_S}(|E_j| + |S_j||T| + |S_j||S_j + T|))$=$O(|E| + |S||T| + |S||N|)$=$O(|N|^2)$ time to identify all L_j's. It takes $O(|S_j||N|^2)$ time to compute the associated $w(L_j)$ for each S-component and identify the node pairs that communicate if a regenerator is added to each node in L_j (this information is later used to update graph B). Thus it takes $O(|S||N|^2)$ time to compute all $w(L_j)$'s and identify L_t^{max}. GH1 then spends $O((|L_t^{max}| + |T|)^2)$ time to add to B edges between nodes that can communicate with each other after a regenerator is placed at each node in L_t^{max}. The iterative steps are repeated for at most $|T|$ times. Thus GH1 takes $O(|T||S||N|^2)$ time in the worst case. □

Proof of Lemma 9: Each **p-for-q** move involves (1) updating the connectivity within the S-components where the changes take place as well as the connectivity of nodes connected to these S-components and (2) checking feasibility of the resulting solution. Updating the adjacency list for each of the neighbors of an s-node takes at most $O(|N|^2)$ time. Therefore, it takes $O(|S||N|^2)$ time to update the adjacency lists, and $O(1)$ time to check for each NDC node pair if the two end points now communicate. The number of NDC node pairs is bounded by $|T|^2$. Thus it takes $O(|S||N|^2 + |T|^2)$ to complete one single **p-for-q** move, independently on the fact if we are performing the move within the same S-component, or between two different ones. □