SIAM J. OPTIM.
Vol. 29, No. 2, pp. 1687–1713

DISTANCE TRANSFORMATION FOR NETWORK DESIGN PROBLEMS*

A. RIDHA MAHJOUB†, MICHAEL POSS‡, LUIDI SIMONETTI§, AND EDUARDO UCHOA¶

Abstract. We propose a new generic way to construct extended formulations for a large class of network design problems with given connectivity requirements. The approach is based on a graph transformation that maps any graph into a layered graph according to a given distance function. The original connectivity requirements are in turn transformed into equivalent connectivity requirements in the layered graph. The mapping is extended to the graphs induced by fractional vectors through an extended linear integer programming formulation. While graphs induced by binary vectors are mapped to isomorphic layered graphs, those induced by fractional vectors are mapped to a set of graphs having worse connectivity properties. Hence, the connectivity requirements in the layered graph may cut off fractional vectors that were feasible for the problem formulated in the original graph. Experiments over instances of the Steiner forest and hop-constrained survivable network design problems show that significant gap reductions compared with state-of-the art formulations can be obtained.

Key words. extended formulations, network design, Benders decomposition

AMS subject classification. 90C10

DOI. 10.1137/16M1108261

1. Introduction.

Let $G = (V, E)$ be a connected and undirected graph with n vertices and m edges with positive costs c_e, $e \in E$. Let $D \subseteq V \times V$ be a set of *demands*. Each demand $(u, v) \in D$ has its own *connectivity requirements*: the existence of a certain number of (u, v)-paths, which may need to satisfy some side constraints. The network design problems (NDPs) considered in this paper consist in finding a subgraph of G with a minimum cost satisfying the connectivity requirements of all demands. We list below a few examples of this class of NDPs.

- Steiner tree problem (STP): the input gives a set $T \subseteq V$ of terminals. D can be defined as $\{(u, v) : u, v \in T,\ u \neq v\}$. The connectivity requirement of a demand $(u, v) \in D$ is the existence of a path joining u and v.
- Steiner forest problem (SFP): an arbitrary set D is given as input. The connectivity requirement is the existence of a path joining the vertices in each demand.
- Hop-constrained Steiner tree problem (HSTP): the input gives a set $T \subseteq V$ of terminals, a root vertex $r \in T$, and an integer H. The demand set D is defined as $\{(r, v) : v \in T \setminus \{r\}\}$. The connectivity requirement is the existence of a path with at most H edges (hops) joining the vertices in each demand. In general, an NDP is said to be rooted if there is a common vertex (the root) for all demands.

*Received by the editors December 16, 2016; accepted for publication (in revised form) April 15, 2019; published electronically June 20, 2019.
http://www.siam.org/journals/siopt/29-2/M110826.html

†Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France (mahjoub@lamsade.dauphine.fr).

‡LIRMM, University of Montpellier, 161 rue Ada, 34095 Montpellier, CNRS, France (michael.poss@lirmm.fr).

§COPPE–PESC, Federal University of Rio de Janeiro, Cidade Universitária, Centro de Tecnologia, Bloco H, CEP 21941-972, Rio de Janeiro, RJ, Brazil (luidi@cos.ufrj.br).

¶Universidade Federal Fluminense, Rua Passo da Pátria 156, CEP 24210-240, Niterói, RJ, Brazil (uchoa@producao.uff.br).

- Hop-constrained survivable network design problem (HSNDP): the set D and integers H and K are given as inputs. The connectivity requirement of a demand $(u, v) \in D$ is the existence of K edge-disjoint (u, v)-paths, each path having at most H edges.

Several other NDPs found in the literature also belong to this class. The connectivity requirement may ask for a number of node-disjoint paths; side constraints may include maximum delay, etc.

On all these cases, the natural formulations over the design variables x_e, $e \in E$, take the following format:

$$\min \quad \sum_{e \in E} c_e x_e \tag{1a}$$

$$\text{s.t. } x \in P(u, v) \quad \forall (u, v) \in D, \tag{1b}$$

$$x_e \text{ binary} \quad \forall e \in E, \tag{1c}$$

where the binary points in polyhedra $P(u, v)$ correspond to all subgraphs satisfying the connectivity requirements of demand (u, v). Even on cases when each polyhedron $P(u, v)$ is integral, i.e., when the formulation already contains the best possible inequalities that are valid for each individual demand, the overall formulation is often still weak.

For example, consider the classic STP, where the natural formulation (1) corresponds to the one proposed by Aneja [1]. In that case, constraints (1b) are given by the so-called undirected cut constraints: for each cut separating two terminals u and v, there must be at least one edge in the solution. The linear relaxation of Aneja's formulation does not provide very good bounds. Even when the formulation is reinforced with sophisticated cuts discovered after polyhedral investigation [4, 5] the duality gaps may be still significant. Fortunately, Wong [23] proposed a much better STP formulation. The main observation is that the set of demands $D = \{(u, v) : u, v \in T,\ u \neq v\}$ can be replaced by $D' = \{(r, v) : v \in T \setminus \{r\}\}$, where $r \in T$ is an arbitrary terminal chosen to be the root. Then, G can be replaced by a bidirected graph $G' = (V, A)$, where A has two opposite arcs (i, j) and (j, i) for each edge $e = \{i, j\}$ in E, with $c((i, j)) = c((j, i)) = c(e)$. The problem now consists in looking for a minimum cost subgraph of G' having a directed path from r to every other terminal. The proposed formulation uses directed cut constraints: for each directed cut separating r from a terminal, there must be at least an arc in the solution. Wong's formulation is remarkably strong. Its duality gaps are less than 0.1% on almost all SteinLib instances [14], except on the artificial instances created with the intent of being hard. In fact, some of the most effective STP codes of today, which are capable of solving nonartificial instances (like those from VLSI design) with tenths of thousands of vertices, usually do not bother to separate cuts other than the simple directed cut constraints. Instead, their algorithmic effort is focused on devising graph reductions and dual ascent procedures in order to speedup the solution of that linear relaxation [19, 20].

Of course, one would like to have similar reformulation schemes able to produce strong bounds for other NDPs. An advance in that direction was the work by Gouveia, Simonetti, and Uchoa [10] on hop-/diameter-constrained spanning/Steiner tree problems. In all those cases it was possible to show that the problem could be transformed into an STP over a directed graph composed of up to H layers, with each layer consisting of copies of the original graph. Additional arcs joining the layers and a few

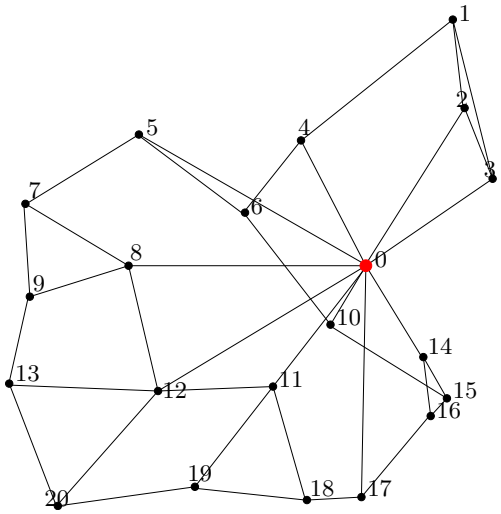


FIG. 1. *Optimal solution of an HSNDP instance with* $K = 3$ *and* $H = 3$*;* $n = 21$*,* $D = \{(0, v) : v = 1, \ldots, 20\}$*, complete graph, Euclidean costs.*

extra vertices/arcs are required. By applying Wong's directed cut formulation over the transformed graphs, very small gaps are obtained. Even though the layered graph approach multiplies the size of the graph, it still allows the solution of instances with hundreds of vertices. Recently, the layered graph approach was successfully applied on a number of other NDPs [21, 16, 8, 9]; effective techniques were introduced to cope with the large transformed graphs and to handle other kinds of constraints.

Nevertheless, it should be noted that all these strong reformulation schemes are restricted to relatively simple NDPs. As the reformulations depend on transforming the problem into a directed STP, they are restricted to problems where the optimal solution topology should be a single tree. NDPs with more complex topologies do not seem to admit directed formulations, for the reasons discussed below.

- Sometimes this is related to the fact that both orientations of an edge can be used in the paths connecting different demands. Figure 1 shows the optimal solution of an HSNDP instance with demand set $D = \{(0, v) : v = 1, \ldots, 20\}$, $K = 3$, and $H = 3$. Even though the instance is rooted at vertex 0, which would make it a natural source for all paths, it can be seen that the same edge can be used in different directions by different demands. For example, demand $(0, 7)$ is connected by paths $0-5-7$, $0-8-9-7$, and $0-12-8-7$; demand $(0, 5)$ is connected by paths $0-5$, $0-10-6-5$, and $0-8-7-5$. Suppose that G is transformed into a bidirected graph. The corresponding directed solution would need to pay for both arcs $(5, 7)$ and $(7, 5)$, which is not correct.
- Even the seemingly simple SFP, where the solution may be formed by a set of disconnected subtrees, does not seem to admit a strong reformulation by turning it into a directed problem over a transformed graph. Here, the difficulty lies in the fact that it is not possible to know beforehand which demands will be connected by the same subtree.

The distance transformation (DT) is an original reformulation technique proposed to obtain stronger formulations for general NDPs, including those where the known

reformulation techniques do not seem to work. Starting from formulation (1), the resulting reformulation will have the following format:

$$\min \quad \sum_{e \in E} c_e x_e \tag{2a}$$

$$\text{s.t. } Ax + Bw + Cy \geq b, \tag{2b}$$

$$(w, y) \in P'(u, v) \quad \forall (u, v) \in D, \tag{2c}$$

$$x_e \text{ binary} \quad \forall e \in E, \tag{2d}$$

where A, B, C, and b are matrices of appropriate dimensions. Constraints (2b) are problem-independent. They are devised to transform a solution over the variables x into a *distance expanded* solution over the new variables w and y. The original connectivity constraints in each $P(u, v)$, over the x variables, are transformed into equivalent connectivity constraints $P'(u, v)$, over the (w, y) variables.

The purpose of the DT can be described informally as follows. Let x be a *binary* vector in $\{0, 1\}^m$ and $G(x) = (V, E(x))$ be the subgraph induced by x. The distance transformation maps $G(x)$ into a subgraph of the layered graph that consists of $n + 1$ copies of G plus edges between adjacent layers. The mapping considers a chosen source subset $S \subseteq V$ and sends each vertex $i \in V$ to a vertex in layer l, $0 \leq l \leq n$, according to its distance from set S. Each edge $\{i, j\} \in G(x)$ is sent to an edge in the layered graph according to the layers of i and j. The transformed graph is represented by variables w and y; the mapping is encoded in constraints (2b). The interest of the transformation is the following. If x is integral, the transformed graph is isomorphic to $G(x)$. However, its extension to graphs $G(x)$ induced by *fractional* vectors leads to transformed graphs that are *less connected* than $G(x)$. Hence, the corresponding (w, y) fractional solution is much more likely to be cut by inequalities (2c).

At first glance, this concept may seem similar to the layered formulations used in [10, 21, 16, 8, 9]. They are, however, completely different. In those previous works, the role of the layered graphs was to model connectivity requirements with hop/distance/delay constraints. In contrast, the DT is not linked to any particular kind of connectivity requirements. In fact, those techniques are orthogonal: it is possible to combine the DT and layered formulations.

1.1. Contributions and structure of the paper. The idea behind DT has its origin in an extended formulation (denoted DT-HOP MCF) introduced in Mahjoub, Simonetti, and Uchoa [18] for the HSNDP. The numerical results obtained were very positive: significant gap reductions with respect to previous known formulations led to the solution of several open instances from the literature. There was, however, no theoretical justification for the gap reductions obtained by what seemed to be an ad-hoc reformulation for that specific network design problem.

The purpose of this paper is to introduce the distance transformation, a generic graph transformation that works for the aforementioned class of network design problems. We show in subsection 2.1 how the distance transformation is naturally defined as a graph transformation. The transformation is extended to graphs defined by fractional vectors by using convexity arguments. We then explain in subsection 2.2 how this ideal distance transformation reduces the connectivity of fractional vectors by splitting nodes. Section 3 studies linear programming formulations for the distance transformation. A natural formulation is presented in subsection 3.1. Subsection 3.2 studies relaxations of the formulation to make it computationally more effective. Section 4 shows how basic connectivity requirements can be incorporated into the linear

programming formulations through inequalities or flow formulations. The resulting formulations are illustrated numerically in section 5 on two network design problems: the Steiner forest problem and the hop-constrained survivable network design problem. Our conclusions are given in section 6.

1.2. Additional notation. Let $F(0,1) = [0,1]^m \setminus \{0,1\}^m$ be the set of vectors in the unit hypercube having at least one fractional component. Undirected edges are denoted by $\{i,j\}$, and directed edges are denoted by (i,j). In addition, we use the following conventions in all figures in the paper. The source set S is indicated by nodes filled in black. When depicting fractional solutions, solid edges and nodes represent edge and node variables with value 1, while dotted edges and nodes represent variables with value 0.5. The set of extreme points of a polytope P is denoted as $\text{ext}(P)$.

2. The unitary distance transformation. We introduce the distance transformation formally in subsection 2.1, and study in subsection 2.2 the connectivity of the layered graphs obtained for fractional vectors x. Linear programming formulations for the DT will be discussed in the next section.

2.1. Definition. The distance transformation considers an arbitrary subset of source nodes $S \subset V$ (even if the problem is rooted, the root may or may not belong to S) and maps any (not necessarily connected) subgraph $G' = (V, E')$ of $G = (V, E)$ to a layered graph. The mapping is based on the distance in G' between any node $i \in V$ and S. Define the layered and undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = V^0 \cup \cdots \cup V^n$, with $V^l = \{i^l : i \in V\}$ for each $l = 0, \ldots, n$ and let i^l be the copy of vertex i in the lth level of graph $\mathbf{G}$. Then, the edge set is defined by

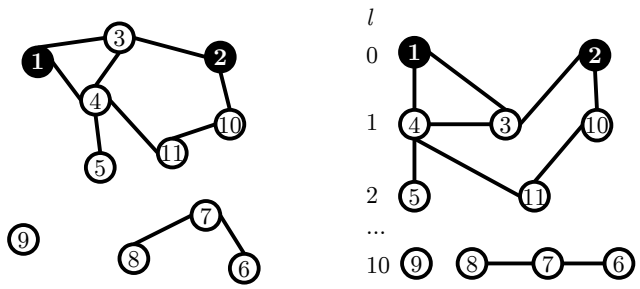
$$\begin{aligned}\mathbf{E} = &\{\{i^l, j^l\} \mid \{i,j\} \in E,\ l \in \{0, \ldots, n-2, n\}\}\\ &\cup \{\{i^l, j^{l+1}\} \mid \{i,j\} \in E,\ 0 \le l \le n-2\}\\ &\cup \{\{i^{l+1}, j^l\} \mid \{i,j\} \in E,\ 0 \le l \le n-2\}.\end{aligned}$$

Let $\mathcal{P}(G)$ be the set of all subgraphs of G that contain all nodes of V and $\mathcal{P}(\mathbf{G})$ be the set of all subgraphs of $\mathbf{G}$ that contain exactly one copy of each node of V, formally: $\mathcal{P}(G) = \{G' = (V, E') \mid E' \subseteq E\}$ and

$$\mathcal{P}(\mathbf{G}) = \{\mathbf{G}' = (\mathbf{V}', \mathbf{E}') \mid \mathbf{V}' \subseteq \mathbf{V} \text{ and } |\mathbf{V} \cap \{i^l : 0 \le l \le n\}| = 1\ \forall i \in V,\ \mathbf{E}' \subseteq \mathbf{E}(\mathbf{V}')\}.$$

The distance transformation is a function $\mathcal{DT}_S : \mathcal{P}(G) \to \mathcal{P}(\mathbf{G})$ that maps any subgraph $G' \in \mathcal{P}(G)$ to layered graph $\mathcal{DT}_S(G') \in \mathcal{P}(\mathbf{G})$ defined as follows. Let $\text{dist}_{G'}(i,S)$ be the length of the shortest path in G' connecting i to a node in S or ∞ if no such path exists. For each $i \in V$ such that $\text{dist}_{G'}(i,S) < \infty$, vertex $i^{\text{dist}_{G'}(i,S)}$ belongs to $\mathcal{DT}_S(G')$. Otherwise, i^n belongs to $\mathcal{DT}_S(G')$. For any $\{i,j\} \in G'$, $\{i^{l_1}, i^{l_2}\}$ belongs to $\mathcal{DT}_S(G')$ if and only if $l_1 = \text{dist}_{G'}(i,S)$ and $l_2 = \text{dist}_{G'}(j,S)$. Note that, for any $\{i,j\} \in E'(G')$, $|\text{dist}_{G'}(i,S) - \text{dist}_{G'}(j,S)| \le 1$. In other words, the images of adjacent vertices in G' lie in adjacent levels in $\mathcal{DT}_S(G')$. Figure 2 shows an example of distance transformation. A crucial property of the distance transformation is that G' and $\mathcal{DT}_S(G')$ are isomorphic.

Any graph in $\mathcal{P}(G)$ can be characterized as $G(x) = (V, E(x))$, where $x \in \{0,1\}^m$ is a binary vector whose component $\{i,j\}$ is equal to 1 if and only if $\{i,j\} \in E(x)$; we say that $G(x)$ is induced by x. In what follows, we denote the binary variable related to edge $e = \{i,j\}$ interchangeably by x_e and x_{ij}. We can also describe a



(a) G'. (b) $\mathcal{DT}_S(G')$.

FIG. 2. *Original graph G' and $\mathcal{DT}_S(G')$ with $S = \{1, 2\}$.*

graph $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ in $\mathcal{P}(\mathbf{G})$ using vectors $w \in \{0,1\}^{n(n+1)}$ and $y \in \{0,1\}^{(3n-2)m}$, defined as follows:

- $w_i^l = 1$ iff $i^l \in \mathbf{V}'$;
- $y_{ij}^{l_1 l_2} = 1$ iff $\{i^{l_1}, j^{l_2}\} \in \mathbf{E}'$.

Conversely, given binary vectors w and y, $\mathbf{G}(w, y)$ denotes the subgraph of $\mathbf{G}$ that contains the vertices (resp., edges) associated to the components of w (resp., y) equal to 1. Hence, we also define the function $DT_S^{int} : \{0,1\}^m \to \{0,1\}^{n(n+1)+(3n-2)m}$ as $DT_S^{int}(x) = (w, y)$, where (w, y) characterizes the graph $\mathcal{DT}_S(G(x))$.

In this paper, we address network design problems formulated as linear programs with binary variables. In this context, x is a vector of optimization variables with values between 0 and 1. To use the distance transformation, we must describe it through a system of linear constraints such that, for each $x \in \{0,1\}^m$, (x, w, y) is feasible if and only if $(w, y) = DT_S^{int}(x)$. Certainly, the smallest polytope defined by such constraints is

$$P_S = \text{conv}\{(x, DT_S^{int}(x)),\ x \in \{0,1\}^m\}. \tag{3}$$

Using P_S, we are able to extend the definition of DT_S^{int} to any vector in $x \in [0,1]^m$, fractional or not, as the following projection:

$$DT_S(x) = \{(w, y) \mid (x, w, y) \in P_S\}. \tag{4}$$

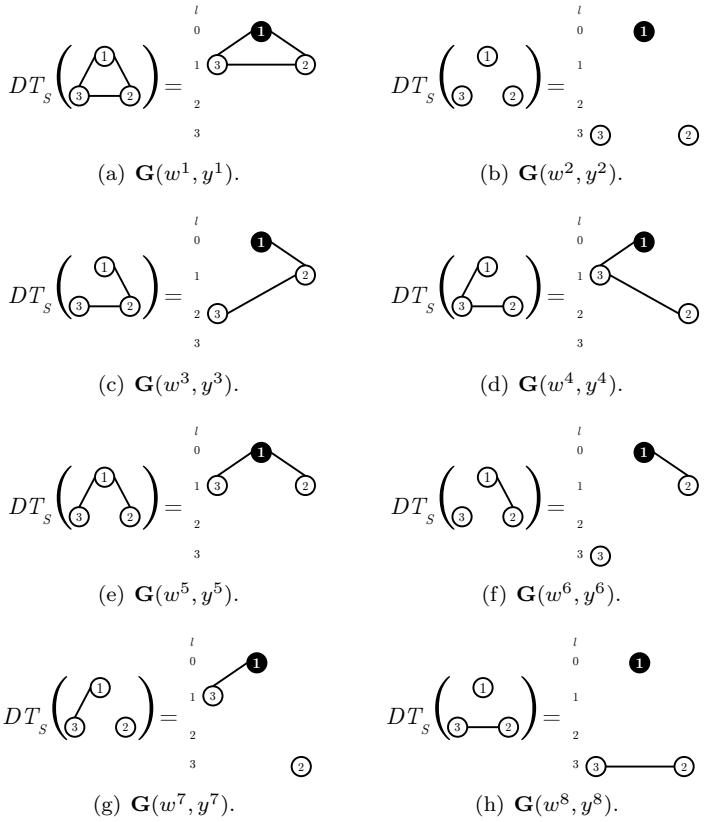
The more general definition (4) is compatible with the previous definition for integral vectors because $DT_S(x)$ reduces to the singleton $\{DT_S^{int}(x)\}$ whenever x is integral. Otherwise, the set $DT_S(x)$ is a polytope with nonzero dimension. Therefore, $DT_S : [0,1]^m \to [0,1]^{n(n+1)+(3n-2)m}$ is a point-to-set mapping. Next, we introduce a characterization of $DT_S(x)$.

THEOREM 1. *Let x^q, $q = 1, \ldots, 2^m$, be the enumerated set of all vectors in $\{0,1\}^m$. A vector (w, y) belongs to $DT_S(x)$ if and only if there exists a vector of convex multipliers λ such that $x = \sum_{q=1}^{2^m} \lambda^q x^q$ and $(w, y) = \sum_{q=1}^{2^m} \lambda^q DT_S^{int}(x^q)$.*

Proof. From (3) and (4), $(w, y) \in DT_S(x)$ if and only if

$$(x, w, y) \in \text{conv}\{(x^q, DT_S^{int}(x^q)),\ 1 \le q \le 2^m\}.$$

So, by the definition of convexity, this is true if and only if there exist convex multipliers λ such that $(x, w, y) = (\sum_{q=1}^{2^m} \lambda^q x^q, \sum_{q=1}^{2^m} \lambda^q DT_S^{int}(x^q))$. ∎



(a) $\mathbf{G}(w^1, y^1)$. (b) $\mathbf{G}(w^2, y^2)$.

(c) $\mathbf{G}(w^3, y^3)$. (d) $\mathbf{G}(w^4, y^4)$.

(e) $\mathbf{G}(w^5, y^5)$. (f) $\mathbf{G}(w^6, y^6)$.

(g) $\mathbf{G}(w^7, y^7)$. (h) $\mathbf{G}(w^8, y^8)$.

FIG. 3. *Graph induced by* $DT_S^{int}(x)$ *for each* $x \in \{x^1, \ldots, x^8\} = \{0,1\}^3$.

Extending the previous definitions to fractional vectors, we can define $G(x)$ and $\mathbf{G}(w, y)$ as the graphs induced by the positive components of x and (w, y), respectively. We provide next an example of $DT_S(\tilde{x})$ for the fractional vector $\tilde{x} = (0.5, 0.5, 0.5)$ associated to the graph $G' = (V', E')$ with $V' = \{1, 2, 3\}$, $E' = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$, and $S = \{1\}$. Figures 3(a) to (h) represent the graphs induced by the vectors of $DT_S(x)$ for each $x \in \{0,1\}^3 = \{x^1, \ldots, x^8\}$. In view of Theorem 1, we obtain

$$(5)\quad DT_S(\tilde{x}) = \left\{ \sum_{q=1}^{8} \lambda^q DT_S^{int}(x^q) \;\middle|\; 0 \le \lambda \le 1,\ \sum_{q=1}^{8} \lambda^q = 1,\ \sum_{q=1}^{8} x_e^q \lambda^q = 0.5,\ e \in E' \right\}.$$

Some of the extreme points of the polytope $DT_S(\tilde{x})$, obtained by setting four of the components of λ to 0, are depicted on Figure 4. One can see the splitting of nodes occurring in the graphs depicted in Figure 4. The splitting of nodes corresponds to fractional values of w. As will be seen in the next section, *the splitting of nodes for all vectors of* $\mathrm{ext}(DT_S(\tilde{x}))$ *is a necessary condition for the DT to be useful.*

This does not always happen. Consider a similar example, except that $S = \{1, 2\}$. The graphs corresponding to some vectors in $\mathrm{ext}(DT_S(\tilde{x}))$ are depicted in Figure 5. We see that the left-hand graph does not contain split nodes. Hence, that choice of S would not give a useful transformation.

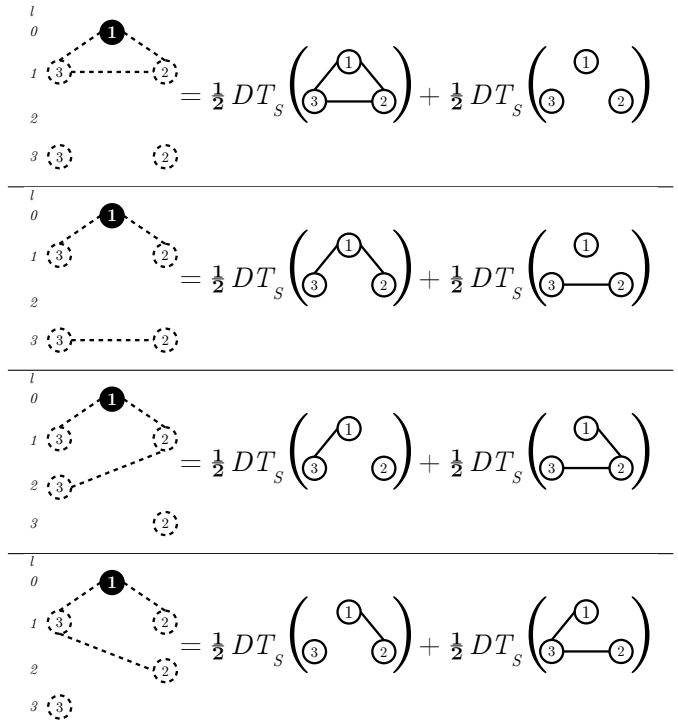


FIG. 4. *Examples of graphs induced by the elements of* $\mathrm{ext}(DT_S(\tilde{x}))$ *when* $S = \{1\}$.

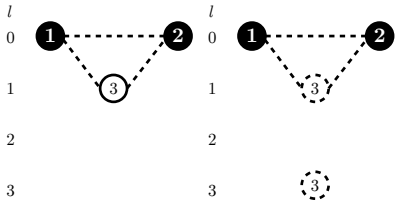


FIG. 5. *Examples of graphs induced by the elements of* $\mathrm{ext}(DT_S(\tilde{x}))$ *when* $S = \{1, 2\}$.

2.2. Connectivity of DT_S. The DT translates the original network design problem defined in G into a network design problem defined in $\mathbf{G}$, by importing to $\mathbf{G}$ the connectivity requirements described in G. In this section, we illustrate how this is done for the simpler case of demands requiring the existence of K edge-disjoint paths between some pairs of vertices. To this end, we add up to $2|D|$ supervertices to $\mathbf{G}$ and $\mathbf{G}(w, y)$, obtaining $\overline{\mathbf{G}} = (\overline{\mathbf{V}}, \overline{\mathbf{E}})$ and $\overline{\mathbf{G}}(w, y) = (\overline{\mathbf{V}}(w), \overline{\mathbf{E}}(w, y))$, respectively. Namely, for each demand $(u, v) \in D$, we create two supervertices $s(u)$ and $t(v)$, respectively linked to vertices u^l and v^l by *directed* edges $(s(u), u^l)$ and $(v^l, t(v))$ for each $0 \le l \le n$. For any $x \in \{0, 1\}^m$, the requirement of the existence in $G(x)$ of K edge-disjoint (u, v)-paths becomes the requirement of the existence in $\overline{\mathbf{G}}(w, y)$ of K edge-disjoint $(s(u), t(v))$-paths. The example from Figure 6 illustrates why the edges connecting the supervertices to $\mathbf{G}$ must be directed, since otherwise we can send 1 unit flow from $s(1)$ to $t(2)$ (0.5 units going through $t(3)$).

To model these connectivity requirements by linear constraints, we need to introduce capacities for the edges of $G(x)$ and $\overline{\mathbf{G}}(w, y)$. The capacity on any edge in

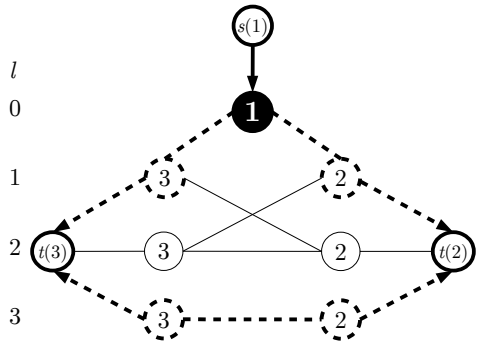


FIG. 6. *Graph* $\overline{\mathbf{G}}(w,y) = (\overline{\mathbf{V}}(w), \overline{\mathbf{E}}(w,y))$ *associated to the second extreme point* (w,y) *depicted in Figure* 4 *and considering demands* $(1,2)$ *and* $(1,3)$. *Thin edges (resp., nodes) correspond to the components of* y *(resp.,* w*) equal to zero and do not belong to* $\overline{\mathbf{E}}(w,y)$ *(resp.,* $\overline{\mathbf{V}}(w)$*).*

$G(x)$ is equal to the value of the associated component of x. For $\overline{\mathbf{G}}(w,y)$, we must distinguish between the undirected edges and the directed ones that link $\mathbf{G}(w,y)$ to the supervertices. The capacity on any undirected edge is equal to the value of the associated component of y, while the capacity on any directed edge linking $s(u)$ (resp., $t(v)$) and u^l (resp., v^l) is equal to Kw_u^l (resp., Kw_v^l). We define next the *connectivity* of the graph $G(x)$ as the vector $C(x) \in \mathbb{R}_+^{|D|}$ with $C_{uv}(x)$ equal to the maximum flow between u and v in $G(x)$. Similarly, we define the connectivity of the graph $\overline{\mathbf{G}}(w,y)$ as the vector $C(w,y) \in \mathbb{R}_+^{|D|}$ with $C_{uv}(w,y)$ equal to the maximum flow between the supervertices $s(u)$ and $t(v)$ in $\overline{\mathbf{G}}(w,y)$. With these definitions, the network design problem defined in G with optimization variables x and connectivity requirements

$$C_{uv}(x) \geq K \quad \forall (u,v) \in D \tag{6}$$

can be reformulated as a network design problem defined in $\overline{\mathbf{G}}$ with optimization variables x, w, and y, and containing two groups of constraints:

1. constraints specifying that $(x,w,y) \in P_S$,
2. the connectivity requirements constraints

$$C_{uv}(w,y) \geq K \quad \forall (u,v) \in D. \tag{7}$$

In the following we denote the feasibility set of constraints (6) by

$$\mathcal{C}^0 = \{x \in [0,1]^m \mid C_{uv}(x) \geq K \ \forall (u,v) \in D\}$$

and the projected feasibility set of constraints (7) by

$$\mathcal{C}^{P_S} = \mathrm{Proj}_x\{(x,y,w) \in P_S \mid C_{uv}(w,y) \geq K \ \forall (u,v) \in D\},$$

where $\mathrm{Proj}_x(\mathcal{X})$ denotes the projection of set $\mathcal{X}$ on variables x. We denote the convex hull of the feasible solutions of the network design problem defined by connectivity requirements (6) by

$$\mathcal{C}^{opt} = \mathrm{conv}\{x \in \{0,1\}^m \mid C_{uv}(x) \geq K \ \forall (u,v) \in D\}.$$

The result below shows that the approximation of $\mathcal{C}^{opt}$ provided by $\mathcal{C}^{P_S}$ is tighter than the one provided by $\mathcal{C}^0$.

PROPOSITION 2. *For any connectivity requirements constraints of the form* (6) *and any* S*, it always holds that*

$$\mathcal{C}^{opt} \subseteq \mathcal{C}^{P_S} \subseteq \mathcal{C}^0. \tag{8}$$

Proof. Let $x \in \{0,1\}^m$ and $DT_S(x) = (w, y)$. The first inclusion follows from the fact that $G(x)$ and $\mathbf{G}(w, y)$ are isomorphic, and thus $C_{uv}(x) = C_{uv}(w, y)$ for each $(u, v) \in D$. To prove the second inclusion, we first prove that

$$C_{uv}(x) \geq C_{uv}(w, y) \quad \forall (u, v) \in D.$$

Let $(u, v) \in D$ and let g be any vector defining a flow from $s(u)$ to $t(v)$ in $\overline{\mathbf{G}}(w, y)$. Then, we can flatten $\overline{\mathbf{G}}(w, y)$ to obtain a flow f from u to v. Namely, we define the flow f on edge $\{i, j\} \in G(x)$ as the sums of the flows described by g on all $\{i^{l_1}, j^{l_2}\} \in \mathbf{G}(w, y)$. The flows on the directed edges linking the supervertices $s(u)$ and $t(v)$ to $\mathbf{G}$ do not matter since we are only interested in a flow from u to v. It is easy to see that the resulting flow f satisfies the capacity constraints, the balance constraints and conveys the same amount of flow from u to v that g conveys from $s(u)$ to $s(v)$. Therefore, if $x \in \mathcal{C}^{P_S}$, we also have that $x \in \mathcal{C}^0$, proving the inclusion. ∎

The power of the distance transformation lies in its reduction of the connectivity of the graphs induced by $(w, y) \in DT_S(x)$ for fractional vectors $x \in (0,1)^m$. Even when a fractional solution x is not cut by connectivity requirement constraints (6), meaning that $x \in \mathcal{C}^0$, it may well happen that all $(w, y) \in DT_S(x)$ are cut by connectivity requirement constraints (7), implying $x \notin \mathcal{C}^{P_S}$. The next result provides an example of $x \in \mathcal{C}^0 \setminus \mathcal{C}^{P_S}$.

PROPOSITION 3. *Consider the network design problem defined on graph* G' *from Figure* 3 *under the connectivity requirements* $C_{12}(x) \geq 1$ *and* $C_{13}(x) \geq 1$*. It holds that*

$$\mathcal{C}^{P_S} \subset \mathcal{C}^0. \tag{9}$$

Proof. The inclusion follows from Proposition 2. To see that the inclusion is strict, we show that the fractional solution $\tilde{x}$ defined by $\tilde{x}_{12} = \tilde{x}_{23} = \tilde{x}_{13} = 0.5$ belongs to $\mathcal{C}^0 \setminus \mathcal{C}^{P_S}$. Clearly, $\tilde{x} \in \mathcal{C}^0$ since we can use the cycle to send half a unit in each direction for both demands in D. To see that $\tilde{x} \notin \mathcal{C}^{P_S}$, consider any $(\tilde{w}, \tilde{y}) \in DT_S(x)$. We show below that either $C_{12}(\tilde{w}, \tilde{y}) < 1$ or $C_{13}(\tilde{w}, \tilde{y}) < 1$.

First, notice using definition (5), that if $\lambda^2 = \lambda^6 = \lambda^7 = \lambda^8 = 0$, then $(w, y) = \sum_{q=1}^{8} \lambda^q (w^q, y^q) \notin \text{ext}(DT_S(\tilde{x}))$. Therefore, for each $(w, y) \in \text{ext}(DT_S(\tilde{x}))$ we have that $w_2^3 > 0$ or $w_3^3 > 0$, which in turn implies

$$\tilde{w}_2^3 > 0 \text{ or } \tilde{w}_3^3 > 0. \tag{10}$$

What is more, $\sum_{l=0}^{n} \tilde{w}_i^l = 1$ for each $i \in V$. Hence, (10) implies that $\tilde{w}_2^1 + \tilde{w}_2^2 < 1$ or $\tilde{w}_3^1 + \tilde{w}_3^2 < 1$. If $\tilde{w}_2^1 + \tilde{w}_2^2 < 1$ (the case when $\tilde{w}_3^1 + \tilde{w}_3^2 < 1$ is similar), we consider demand $(1, 2)$ and the cut associated with the partition $W = \{t(2), 2^3, 3^3\}$ and $\bar{W} = \overline{\mathbf{V}}(\tilde{w}, \tilde{y}) \setminus W$. The capacity of the cut is equal to $\tilde{w}_2^1 + \tilde{w}_2^2 < 1$, proving $C_{12}(\tilde{w}, \tilde{y}) < 1$. ∎

The intuitive idea behind the distance transformation is that fractional solutions are often mapped to layered graphs where node splitting occurs, as in Figure 4. The node splitting then cuts some paths in the original graph, which results in a

decrease of connectivity. This decrease is often enough to cut the fractional solutions. Figure 4 shows that the node splitting breaks the cycle in all cases but the upper one. Nevertheless, the connectivity of the upper-left graph is also reduced because of the limited capacity available on the directed edges linking the supervertices to the layered graph. When no node splitting occurs, as in the left-hand graph of Figure 5, graphs $G(x)$ and $\mathbf{G}(w, y)$ are isomorphic, so they satisfy the same connectivity requirements. It is therefore useful to be able to discover whether a given distance transformation leads to splitting of the fractional vectors. The result below partially answers this question by providing an approach to find out whether the graph induced by a given fractional solution $(x, w, y) \in P_S$ contains split nodes.

PROPOSITION 4. *Let $x \in F(0,1)$ and $(w, y) \in DT_S(x)$. Consider a set of Q positive convex multipliers λ such that*

$$(x, w, y) = \sum_{q=1}^{Q} \lambda^q (x^q, w^q, y^q), \tag{11}$$

where $x^q \in \{0,1\}^m$ and $(w^q, y^q) = DT_S(x^q)$ for each $q = 1, \ldots, Q$. Then, any node $i \in V$ is split Q' times in $\mathbf{G}(w, y)$, where $Q' \in \{1, \ldots, Q\}$ corresponds to the number of different values in set

$$\{\mathrm{dist}_{G(x^q)}(i, S),\ q = 1, \ldots, Q\}. \tag{12}$$

Proof. From (11), we have that

$$w_i^l = \sum_{q=1}^{Q} \lambda^q w_i^{ql}.$$

By the definition of DT_S, we further have that $w_i^{q\,\mathrm{dist}_{G(x^q)}(i,S)} = 1$ for each $q \in \{1, \ldots, Q\}$, so that

$$w_i^l = \sum_{q=1}^{Q} \lambda^q w^q = \sum_{q:\mathrm{dist}_{G(x^q)}(i,S)=l} \lambda^q,$$

which is positive for each $l \in \{0, \ldots, n\}$ corresponding to a value in the set defined in (12). ∎

Proposition 4 is an important result in understanding the structure of fractional vectors in P_S, which is the key to efficiently apply the DT to network design problems. In particular, the proposition shows that any node $i \in V$ corresponding to (x, w, y) defined by (11) is split if and only if

$$\mathrm{dist}_{G(x^q)}(i, S) \neq \mathrm{dist}_{G(x^{q'})}(i, S) \tag{13}$$

for some $q \neq q'$ in $\{1, \ldots, Q\}$. The difficulty of using the node splitting to cut a particular fractional solution x is that we must ensure that the splitting occurs for all $(w, y) \in DT_S(x)$, which is a complex task in general. Nevertheless, for some very special cases it is possible to predict that splitting always occurs (see the result below).

PROPOSITION 5. *Consider the DT defined by a unitary source $S = \{i\}$ for some $i \in V$, let $x \in F(0,1)$, and consider a node $j \in V \setminus \{i\}$. If $x_{ij} \in (0,1)$, node j is split in $\mathbf{G}(w, y)$ for all $(w, y) \in DT_S(x)$.*

Proof. Let

$$(x, w, y) = \sum_{q=1}^{Q} \lambda^q (x^q, w^q, y^q)$$

be any vector in P_S. Because $x_{ij} \in (0, 1)$, there exist q and q' in Q such that $x^q_{ij} = 1$ and $x^{q'}_{ij} = 0$. Hence, (13) holds, yielding the result. ∎

3. Linear programming formulations for the DT.

3.1. "Natural" formulation for P_S. So far we have been using abstract distance transformations based on the ideal polytope P_S. To use DT in practice, we need a linear formulation for P_S, providing its convex hull in the ideal case. Unfortunately, the complete description of P_S is not easy to find. Below, we provide a polynomial formulation for P_S which, although it does not completely describe P_S, leads to very good improvements in the linear programming relaxation of some network design problems with connectivity requirements. Let $\delta_E(S) = \{\{i, j\} \in E,\ |\{i, j\} \cap S| = 1\}$ and $E' = E \setminus \delta_E(S)$. The formulation below links the three groups of variables x, w, and y using the following constraints:

$$w^0_i = 1 \qquad \forall i \in S, \tag{14a}$$

$$\sum_{l=1}^{n} w^l_i = 1 \qquad \forall i \in V \setminus S, \tag{14b}$$

$$y^{01}_{ij} = x_{ij} \qquad \forall \{i, j\} \in \delta_E(S) : i \in S, \tag{14c}$$

$$\sum_{l=1}^{n} y^{ll}_{ij} + \sum_{l=1}^{n-2} (y^{l(l+1)}_{ij} + y^{l(l+1)}_{ji}) = x_{ij} \qquad \forall \{i, j\} \in E', \tag{14d}$$

$$y^{01}_{ij} \le w^1_j \qquad \forall \{i, j\} \in \delta_E(S) : i \in S, \tag{14e}$$

$$\begin{aligned} y^{11}_{ij} + y^{12}_{ij} &\le w^1_i, \\ y^{11}_{ij} + y^{12}_{ji} &\le w^1_j \end{aligned} \qquad \forall \{i, j\} \in E', \tag{14f}$$

$$\begin{aligned} y^{ll}_{ij} + y^{l(l+1)}_{ij} + y^{(l-1)l}_{ji} &\le w^l_i, \\ y^{ll}_{ij} + y^{l(l+1)}_{ji} + y^{(l-1)l}_{ij} &\le w^l_j \end{aligned} \qquad \forall \{i, j\} \in E',\ l = 2, \dots, n-2, \tag{14g}$$

$$\begin{aligned} y^{(n-2)(n-1)}_{ji} + y^{(n-1)(n-1)}_{ji} &\le w^{n-1}_i, \\ y^{(n-2)(n-1)}_{ij} + y^{(n-1)(n-1)}_{ij} &\le w^{n-1}_j \end{aligned} \qquad \forall \{i, j\} \in E', \tag{14h}$$

$$w^l_i \le \sum_{\{j,i\} \in E} y^{(l-1)l}_{ji} \qquad \forall i \in V \setminus S,\ l = 1, \dots, n-1, \tag{14i}$$

$$0 \le x, w, y \le 1. \tag{14j}$$

Constraints (14a) and (14b) state that each vertex should be in one of its possible levels. Constraints (14c)–(14h) state that each original edge variable x_{ij} should be translated into a variable $y^{l_1 l_2}_{ij}$ such that both $w^{l_1}_i$ and $w^{l_2}_j$ have value 1. Constraints (14i) state that a vertex i can only be in level $l < n$ if it is reached by at least one edge $\{j, i\}$ from level $(l - 1)$. Let P^F_S be the polytope defined by constraints (14). We prove below that the above constraints yield a valid formulation for the set $\{(x, DT^{int}_S(x)) : x \in \{0, 1\}^m\}$.

PROPOSITION 6. *The linear constraints in* (14) *yield a valid formulation for* $X = \{(x, DT_S^{int}(x)) : x \in \{0,1\}^m\}$. *Hence, given* $x \in \{0,1\}^m$, $(x,w,y) \in P_S^F$ *if and only if* $(w,y) = DT_S^{int}(x)$.

Proof. The validity of (14a)–(14j) follows from the observation that $(x, DT_S^{int}(x))$ satisfies the constraints for each $x \in \{0,1\}^m$. To see that (14a)–(14j) are also a formulation for X, we consider a binary vector $x \in \{0,1\}^m$ and prove next by induction on $k = 0, \ldots, n-1$ that P_S^F contains a unique integer solution where, for each node i connected to S in $G(x)$,

$$w_i^k = 1 \Leftrightarrow \text{dist}_{G(x)}(S,i) = k. \tag{15}$$

The case when $k = 0$ is immediate from (14a), while the case when $k = 1$ follows from (14c), (14e), and (14i).

Consider $k \in \{2, \ldots, n-2\}$ (the proof for $k = n-1$ is similar) and suppose (15) is true for $l \in \{0, \ldots, k-1\}$. Let $j' \in V$ be such that $\text{dist}_{G(x)}(j', S) = k$. There exists an edge $\{i', j'\} \in E(x)$ such that $\text{dist}_{G(x)}(i', S) = k-1$. Hence, by induction, $w_{i'}^{k-1} = 1$. Because $w_{i'}^l = 0$ for each $l \neq k-1$, the first inequalities of (14f) and (14g) imply that $y_{i'j'}^{l_1 l_2} = 0$ if $l_1 \neq k-1$, so (14d) written for $\{i', j'\}$ becomes

$$y_{i'j'}^{(k-1)(k-1)} + y_{i'j'}^{(k-1)k} + y_{j'i'}^{(k-2)(k-1)}.$$

Then, because $w_{i'}^l = 0$ for each $l \leq k-1$, the second set of inequalities of (14g) implies $y_{j'i'}^{(k-2)(k-1)} = 0$ and $y_{i'j'}^{(k-1)(k-1)} = 0$, so $y_{i'j'}^{(k-1)k} = 1$. Hence, the second inequality of (14g) written for $\{i', j'\}$ and $l = k$ yields $w_{j'}^k = 1$.

Reciprocally, suppose $w_{j'}^k = 1$. By induction,

$$\text{dist}_{G(x)}(j', S) \geq k.$$

What is more, (14i) implies that there exists $\{i', j'\}$ such that $y_{i'j'}^{(k-1)k} = 1$. Therefore, $w_{i'}^{k-1} = 1$, so, by induction, $\text{dist}_{G(x)}(i', S) = k-1$, which in turn implies $\text{dist}_{G(x)}(j', S) \leq k$.

If i is not connected to S, then the corresponding (14b) constraint ensures that $w_i^n = 1$. ∎

The above formulation enables us to extend DT_S to fractional vectors, as done in subsection 2.1. The counterpart of (4) for P_S^F is

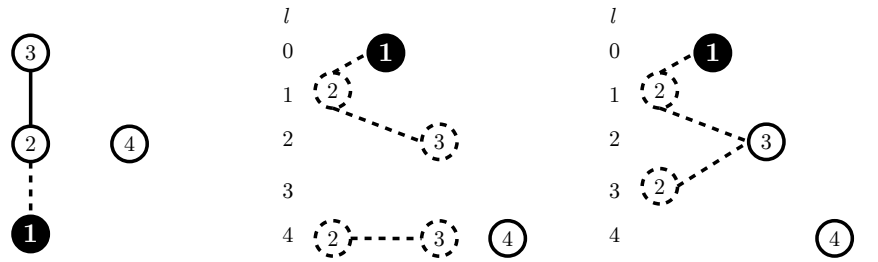
$$DT_S^F(x) = \{(w,y) \mid (x,w,y) \in P_S^F\}, \tag{16}$$

and $P \subseteq P_S^F$ implies that $DT_S(x) \subseteq DT_S^F(x)$ for any $x \in [0,1]^n$. When x is fractional, the example from Figure 7 shows that the inclusion can be strict. Namely, Figure 7(c) depicts the graph induced by a vector $(w,y) \in DT_S^F(x)$ that does not belong to $DT_S(x)$ because it cannot be obtained as the convex combination of binary vectors. In fact, for that example, $DT_S(x)$ is reduced to the singleton (w,y) that induces the graph in Figure 7(b).

For any $x \in [0,1]^m$, one can also extend the connectivity requirements to the graphs induced by any $(w,y) \in DT_S^F(x)$. We define

$$\mathcal{C}^{F_S} = \text{Proj}_x\{(x,y,w) \in P_S^F \mid C_{uv}(w,y) \geq K \; \forall (u,v) \in D\},$$

and Proposition 2 can be completed with the following result.



(a) $G(x)$. (b) *Graph* $\mathbf{G}(w,y)$ *for the unique* $(w,y) \in DT_S(x)$. (c) *Graph* $\mathbf{G}(w,y)$ *for some* $(w,y) \in DT_S^F(x)$.

FIG. 7. *Example of fractional* x*, the single point in* $DT_S(x)$*, and a point in* $DT_S^F(x)$ $(S = \{1\})$.

PROPOSITION 7. *For any connectivity requirements constraints, it always holds that*

$$\mathcal{C}^{P_S} \subseteq \mathcal{C}^{F_S} \subseteq \mathcal{C}^0. \tag{17}$$

Proof. The inclusion $\mathcal{C}^{P_S} \subseteq \mathcal{C}^{F_S}$ follows from the fact that $P \subseteq P_S^F$. Then, constraints (14d) and (14f) enable us to prove inclusion $\mathcal{C}^{F_S} \subseteq \mathcal{C}^0$ by using the same flattening argument as the one used in the proof of Proposition 2. ∎

We study next whether inclusions in (17) can be strict. One can verify numerically that Proposition 3 extends to $\mathcal{C}^{F_S}$, providing an instance for which $\mathcal{C}^{F_S} \subset \mathcal{C}^0$. We then turn to inclusion $\mathcal{C}^{P_S} \subseteq \mathcal{C}^{F_S}$. The example from Figure 7 does not lead to strict inclusion because the graphs of Figures 7(b) and (c) satisfy the same connectivity requirements, namely $C_{23}(w,y) \geq 1$. However, we do have numerical evidence that the inclusion can be indeed strict, which is left out of the manuscript to simplify our exposure.

3.2. Limiting the levels. The DT described in the previous subsections can lead to large formulations. Due to the number of layers in graph $\mathbf{G}$, formulation (14) introduces $O(nm)$ variables and constraints. For some NDPs, we can eliminate most levels without affecting the transformation. For instance, if all demands in D have a common extremity, and the connectivity requirements ask for paths bounded by some number H, then we can restrict the number of layers to $H+1$. This example arises in the survivable network design problems studied in [18].

However, it is possible to define DTs that use only a small number of layers L, regardless of the NDP under consideration. This decreases the size of the associated linear programming formulations, but may also decrease the node splitting, and thus the gains in terms of gap reduction. In fact, there is a trade-off between the chosen value of L and the quality of the DT. We suppose that $G(x)$ is the graph induced by some binary vector x and that $\mathbf{G}_L$ is a graph that consists of L layers. The truncated distance transformation DT_S^L sets the image of node i in $\mathbf{G}_L$ to layer $\min(\text{dist}_{G(x)}(i,S), L-1)$. This means that levels from $L-1$ to $n-1$ of the original $\mathbf{G}$ are flattened into a single level $L-1$ in $\mathbf{G}_L$; nodes not connected to S are still mapped to layer L. Edges are mapped subsequently according to the images of their extremities. This will not greatly affect the quality of the DT when few nodes i have $\text{dist}_{G(x)}(i,S) \geq L$ in typical solutions x. Formulation (14) is adapted for this

modification by changing constraints (14g)–(14i), reducing the size of the formulation to $O(Lm)$ variables and constraints.

4. Formulating the connectivity requirements.

4.1. Simple connectivity requirements. In order to provide a linear programming formulation for an NDP, one still has to reformulate the connectivity requirements constraints with linear constraints. We start our approach with the simple constraints

$$C_{uv}(w,y) \geq K \quad \forall (u,v) \in D \tag{18}$$

considered in the previous sections. Recall that constraints (18) impose that, for each $(u,v) \in D$, the value of the maximum flow between $s(u)$ and $t(v)$ in $\overline{\mathbf{G}}(w,y)$ is not smaller than K; see Figure 6 for an example of graph $\overline{\mathbf{G}}(w,y) = (\overline{\mathbf{V}}(w), \overline{\mathbf{E}}(w,y))$. We next describe how to express the constraints for a single demand $\{u,v\} \in D$ using either flow variables or cut inequalities and disregarding the level reductions discussed in the previous section. The flow formulation complements the formulation in (14) with two flow variables $f_{ij}^{l_1 l_2}$ and $f_{ji}^{l_2 l_1}$ for each undirected edge $\{i^{l_1}, j^{l_2}\} \in \mathbf{E}$ and two flow variables $f_{s(u)u}^{l}$ and $f_{vt(v)}^{l}$ for each demand $(u,v) \in D$ and $l \in \{1,\dots,n\}$ (notice that if u belongs to S, we only introduce flow variable $f_{s(u)u}^{0}$ (see Figure 6), and similarly if $v \in S$). Then, we impose the conditions that the capacity be respected for all edges

$$\begin{aligned} f_{ij}^{l_1 l_2} + f_{ji}^{l_2 l_1} &\leq y_{ij}^{l_1 l_2} \quad \forall \{i^{l_1}, j^{l_2}\} \in \mathbf{E}, \\ f_{s(u)i}^{l} &\leq K w_i^l \quad \forall i^l \in \mathbf{V}, \\ f_{it(v)}^{l} &\leq K w_i^l \quad \forall i^l \in \mathbf{V}, \end{aligned} \tag{19}$$

the flow be conserved for nodes in $\mathbf{V}$, i.e.,

$$\sum_{\{i^{l_1}, j^{l_2}\} \in \delta_{\mathbf{E}}(i^{l_1})} \left(f_{ji}^{l_2 l_1} - f_{ij}^{l_1 l_2} \right) = 0 \quad \forall i^{l_1} \in \mathbf{V}, \tag{20}$$

adding the term $f_{s(i)i}^{l_1}$ (resp., $f_{it(i)}^{l_1}$) if $i = u$ (resp., $i = v$) for some $(u,v) \in D$, and that the flow exiting supervertex $s(u)$ exceeds K:

$$\begin{aligned} f_{s(u)i}^{0} &\geq K \quad \text{if } u \in S, \\ \sum_{l=1}^{n} f_{s(u)i}^{l} &\geq K \quad \text{otherwise.} \end{aligned} \tag{21}$$

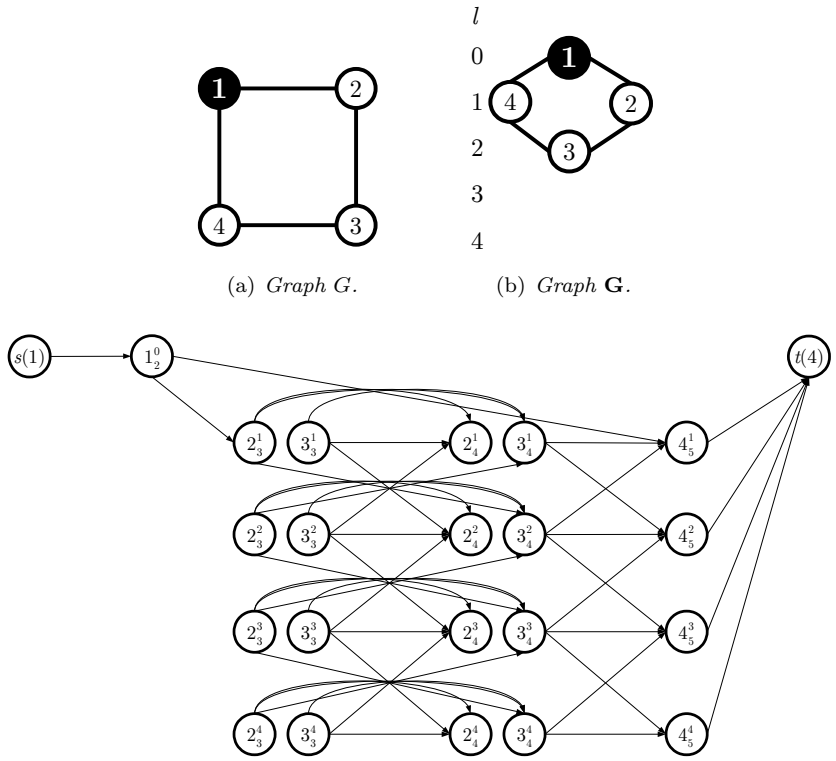
Alternatively to constraints (19)–(21), cut inequalities (see, e.g., [1]) impose the condition that

$$\sum_{l: u^l \in \mathbf{V} \setminus \mathbf{U}} K w_u^l + \sum_{l: v^l \in \mathbf{U}} K w_v^l + \sum_{\{i^{l_1}, j^{l_2}\} \in \delta_{\mathbf{E}}(\mathbf{U})} y_{ij}^{l_1 l_2} \geq K \quad \forall \mathbf{U} \subseteq \mathbf{V}. \tag{22}$$

We next illustrate cut inequalities on an example based on the solution depicted in Figure 6 together with connectivity requirement $C_{12}(w,y) \geq 1$. If $\mathbf{U} = \{1^0, 2^1, 3^1\}$, then inequality (22) becomes

$$w_2^1 + y_{23}^{12} + y_{32}^{12} \geq 1,$$

which is violated by the solution depicted in Figure 6.



(a) *Graph* G. (b) *Graph* $\mathbf{G}$.

(c) *Graph* $\mathbf{G}^{14}$ *for* $H = 3$.

FIG. 8. *Hop-level graph.*

4.2. Hop constraints. We study next the more complex connectivity requirements obtained by limiting the path length (hops) used to transmit the flow by a given integer H. Namely, we consider matrix $\mathfrak{C}(x) \in \mathbb{R}_+^{|D|\times(n-1)}$ with $\mathfrak{C}^H_{uv}(x)$ equal to the maximum flow between u and v in $G(x)$ using paths with at most H hops. Similarly, we define matrix $\mathfrak{C}(w,y) \in \mathbb{R}_+^{|D|\times(n-1)}$ with $\mathfrak{C}^H_{uv}(w,y)$ equal to the maximum flow between the supervertices $s(u)$ and $t(v)$ in $\overline{\mathbf{G}}(w,y)$ using paths with at most $H+2$ hops. With these definitions, we see immediately that $C(x) = \mathfrak{C}^{n-1}(x)$ and $C(w,y) = \mathfrak{C}^{n-1}(w,y)$. The counterparts of (6) and (18) for $\mathfrak{C}$ are

$$\mathfrak{C}^H_{uv}(x) \geq K \quad \forall (u,v) \in D, \tag{23}$$

and

$$\mathfrak{C}^H_{uv}(w,y) \geq K \quad \forall (u,v) \in D. \tag{24}$$

Recall that connectivity constraints (23) can be expressed by a hop-indexed flow formulation first introduced in [7]. In [18] the formulation was extended to handle (24), and was denoted the DT-HOP indexed flow formulation therein. We recall it below. Given a demand $(u,v) \in D$ and an integer H, the formulation considers a directed layered graph $\mathbf{G}^{uv} = (\mathbf{V}^{uv}, \mathbf{A}^{uv})$, where the definition of $\mathbf{V}^{uv} = \mathbf{V}^{uv}_1 \cup \cdots \cup \mathbf{V}^{uv}_{H+3}$ depends on whether $\{u,v\}$ intersects S. If $\{u,v\} \cap S = \emptyset$, then $\mathbf{V}^{uv}_1 = \{s(u)\}$,

$\mathbf{V}_2^{uv} = \{u^l \mid 1 \le l \le n\}$, $\mathbf{V}_h^{uv} = \mathbf{V} \setminus \{\{u^l, v^l\} \mid 1 \le l \le n\}$, $h = 3, \ldots, H+1$, $\mathbf{V}_{H+2}^{uv} = \{v^l \mid 1 \le l \le n\}$, and $\mathbf{V}_{H+3}^{uv} = \{t(v)\}$. If $u \in S$ or $v \in S$, we have instead $\mathbf{V}_2^{uv} = \{u^0\}$ or $\mathbf{V}_{H+2}^{uv} = \{v^0\}$, respectively; see Figure 8 for an example where $u \in S$ and $v \notin S$. Let $\mathbf{i}_h$ be the copy of $\mathbf{i} \in \mathbf{V}$ in the hth layer of graph $\mathbf{G}^{uv}$, that is, $\mathbf{i} = i^l$ for some $i \in V$, $0 \le l \le n$, and $\mathbf{i}_h = i_h^l$. The arc set is defined by (see again Figure 8)

$$\mathbf{A}^{uv} = \{(s(u), \mathbf{u}_2) \mid \mathbf{u}_2 \in \mathbf{V}_2^{uv}\} \tag{25}$$

$$\cup \{(\mathbf{u}_2, \mathbf{i}_3) \mid \{\mathbf{u}, \mathbf{i}\} \in \mathbf{E},\ \mathbf{u}_2 \in \mathbf{V}_2^{uv},\ \mathbf{i}_3 \in \mathbf{V}_3^{uv}\} \tag{26}$$

$$\cup \{(\mathbf{i}_h, \mathbf{j}_{h+1}) \mid \{\mathbf{i}, \mathbf{j}\} \in \mathbf{E},\ \mathbf{i}_h \in \mathbf{V}_h^{uv},\ \mathbf{j}_{h+1} \in \mathbf{V}_{h+1}^{uv},\ 3 \le h \le H\} \tag{27}$$

$$\cup \{(\mathbf{i}_h, \mathbf{i}_{h+1}) \mid 3 \le h \le H,\ \mathbf{i}_h \in \mathbf{V}_h^{uv}\} \tag{28}$$

$$\cup \{(\mathbf{i}_{H+1}, \mathbf{v}_{H+2}) \mid \{\mathbf{v}, \mathbf{i}\} \in \mathbf{E},\ \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv},\ \mathbf{i}_{H+1} \in \mathbf{V}_{H+1}^{uv}\} \tag{29}$$

$$\cup \{(\mathbf{v}_{H+2}, t(v)) \mid \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv}\} \tag{30}$$

$$\cup \{(\mathbf{u}_2, \mathbf{v}_{H+2}) \mid \{\mathbf{u}, \mathbf{v}\} \in \mathbf{E},\ \mathbf{u}_2 \in \mathbf{V}_2^{uv},\ \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv}\}. \tag{31}$$

Given this auxiliary graph, the DT-HOP indexed flow formulation complements formulation (14) with a flow variable $g_{\mathbf{ij}}^{uv,h}$ for each arc $(\mathbf{i}_h, \mathbf{j}_{h+1}) \in \mathbf{A}^{uv}$. To simplify notation, we omit index uv from the flow variables in what follows. Then, we impose that capacity be respected for all edges,

$$g_{s(u)\mathbf{u}}^1 \le K w_{\mathbf{u}} \quad \forall \mathbf{u}_2 \in \mathbf{V}_2^{uv}, \tag{32}$$

$$g_{\mathbf{ui}}^2 \le y_{\mathbf{ui}} \quad \forall \{\mathbf{u}, \mathbf{i}\} \in \mathbf{E},\ \mathbf{u}_2 \in \mathbf{V}_2^{uv},\ \mathbf{i}_3 \in \mathbf{V}_3^{uv}, \tag{33}$$

$$\sum_{h=3}^{H} \left(g_{\mathbf{ij}}^h + g_{\mathbf{ji}}^h\right) \le y_{\mathbf{ij}} \quad \forall \{\mathbf{i}, \mathbf{j}\} \in \mathbf{E},\ \mathbf{i}_h \in \mathbf{V}_h^{uv},\ \mathbf{j}_{h+1} \in \mathbf{V}_{h+1}^{uv},\ 3 \le h \le H, \tag{34}$$

$$g_{\mathbf{iv}}^{H+1} \le y_{\mathbf{vi}} \quad \forall \{\mathbf{v}, \mathbf{i}\} \in \mathbf{E},\ \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv},\ \mathbf{i}_{H+1} \in \mathbf{V}_{H+1}^{uv}, \tag{35}$$

$$g_{\mathbf{v}t(v)}^{H+2} \le K w_{\mathbf{v}} \quad \forall \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv}, \tag{36}$$

$$g_{\mathbf{uv}} \le y_{\mathbf{uv}} \quad \forall \{\mathbf{u}, \mathbf{v}\} \in \mathbf{E},\ \mathbf{u}_2 \in \mathbf{V}_2^{uv},\ \mathbf{v}_{H+2} \in \mathbf{V}_{H+2}^{uv}. \tag{37}$$

Arcs in (28) have an infinite capacity, so no capacity constraints are written for these arcs. The counterpart of flow conservation constraints in graph $\mathbf{G}^{uv}$ is readily obtained from (20). Finally, we need to impose that the flow exiting supervertex $s(u) \in \mathbf{V}_1^{uv}$ exceeds K:

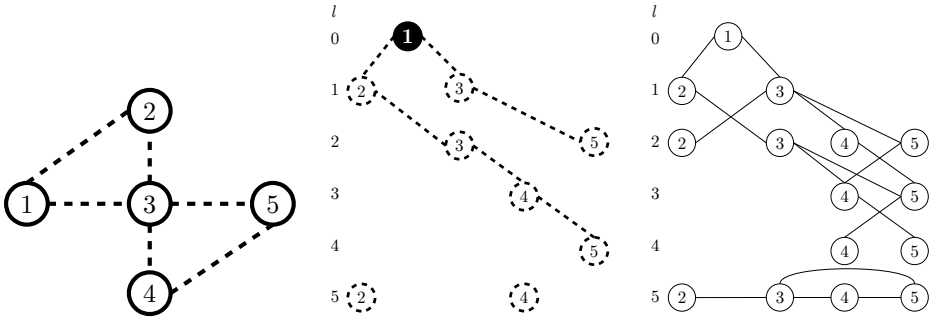
$$\sum_{\mathbf{u}_2 \in \mathbf{V}_2^{uv}} g_{s(u)\mathbf{u}_2}^0 \ge K. \tag{38}$$

Capacity constraints (34) prevent the above system from defining a pure network flow problem that has the integrality property. Hence, in general we need to impose integrality restrictions on g. However, in the cases when $H = 2, 3$, one can readily extend the results from [11, 2] to show that the DT-HOP indexed formulation is indeed integral. Actually, for those cases it is possible to avoid including the DT-HOP indexed variables and constraints in the formulation (14) and to replace them by cuts separated by the min-cut algorithm over $\mathbf{G}^{uv}$, as shown in the following example.

Consider graph $G = (V, E)$, defined by

$$V = \{1, 2, 3, 4, 5\} \quad \text{and} \quad E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}.$$

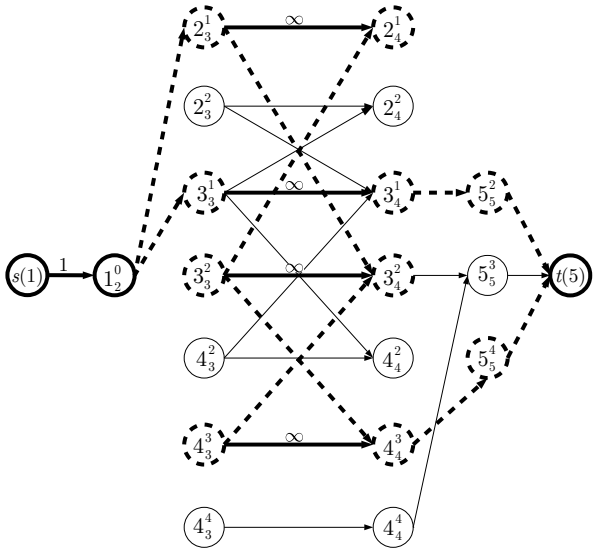
Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be the associated layered graph and Figure 9(a) represent a fractional solution that satisfies $\mathfrak{C}_{15}^3(x) \ge 1$. Figure 9(b) represents the graph associated



(a) *A fractional solution* $G(x)$ *that satisfies* $\mathfrak{C}^3_{15}(x) \geq 1$.

(b) $\mathbf{G}(w,y)$ *for some* $(w,y) \in DT_S(x)$ *that violates the condition* $\mathfrak{C}^3_{15}(w,y) \geq 1$.

(c) $\bigcup_{\tilde{x}\in\{0,1\}^m} \mathbf{G}(DT_S(\tilde{x}))$.



(d) *Transformation of* $\mathbf{G}(w,y)$ *following Figure* 1 *in* [12]. *Thin nodes (resp., edges) correspond to the components of* w *(resp.,* y*) equal to zero. Nodes in level* 5 *are omitted because they are not connected to* 1.

FIG. 9. *Separation of* 3*-path cut inequalities following* [12].

to some $(w,y) \in DT_S(x)$ that violates $\mathfrak{C}^3_{15}(w,y) \geq 1$. We show next how to separate a violated cut violated by $\mathbf{G}(w,y)$. Without loss of generality, we can restrict ourselves to the subsets of nodes and edges of $\mathbf{G}$ that belong to at least one graph of $\{\mathbf{G}(DT(\tilde{x})) : \tilde{x} \in \{0,1\}^m\}$; these are represented in Figure 9(c) and can be obtained automatically using preprocessing algorithms. The feasibility of $\mathfrak{C}^3_{15}(w,y) \geq 1$ is tested by looking for a feasible flow of one unit between $s(1)$ and $t(5)$ in the expanded graph depicted in Figure 9(d). Looking for a cut of minimum capacity that contains $s(1)$, we obtain either the inequality

$$y^{01}_{13} + y^{23}_{35} \geq 1$$

or the inequality

$$w_5^2 + y_{35}^{23} \geq 1,$$

which are both violated by the solution depicted in Figure 9(b). Those inequalities correspond to the counterparts of the 3-path-cut inequalities proposed in [11, 2].

Whenever $H \geq 4$, the DT-HOP indexed flow formulation cannot be replaced by inequalities obtained by the min-cut algorithm. However, it can be numerically efficient to avoid the inclusion of the formulation and replace it by Benders inequalities for variables y and w, similarly to [3].

5. Numerical experiments for the DT.

5.1. The Steiner forest problem.

The SFP has the following natural formulation:

$$\text{(39a)} \qquad \min \quad \sum_{e \in E} c_e x_e$$

$$\text{(39b)} \qquad \text{s.t.} \quad \sum_{e \in \delta_E(S)} x_e \geq 1 \quad \forall (u,v) \in D,\ \forall S \subset V,\ u \in S,\ v \notin S,$$

$$\text{(39c)} \qquad x_e \text{ binary} \quad \forall e \in E.$$

Constraints (39b) are known as undirected cut inequalities. This formulation is the basis for the classical primal-dual 2-approximated algorithm for the SFP [6]. Nevertheless, (39) does not provide effective exact branch-and-cut algorithms; duality gaps of more than 20% are typical on practical instances. For example, consider an instance defined over a complete graph with vertices $\{1, 2, 3\}$ and $D = \{(1, 2), (1, 3)\}$. The fractional solution $x_{12} = x_{13} = x_{23} = 0.5$ mentioned previously satisfies all constraints (39b).

The difficulty of devising a directed formulation for SFP lies in the fact that it is not known beforehand which demands will belong to the same connected component (a subtree) of an optimal solution. A strong SFP formulation was proposed by Magnanti and Raghavan [17], based on the concept of *consistent edge orientations across demands*. In that concept, each connected component is represented by a directed tree rooted at the first vertex of the demand with smaller index in the component. However, the formulation in [17] is not very practical since it contains an exponential number of constraints, and no polynomial algorithm for separating them is known. The lifted-cut formulation proposed by Konemann et al. [15] can be stronger than (39) and its linear relaxation can be solved in polynomial time. Recently, Schmidt et al. [22] presented four new formulations (essentially three formulations, since two of them are equivalent). The strongest new formulations are based on the concept of consistent edge orientations across demands from [17]; however, their linear relaxations can be solved in polynomial time.

Figure 10 depicts the small SFP instance with 8 vertices and 12 edges considered in [22]; edge costs are unitary and the four demands are represented by pairs of identical symbols. The optimal solution value is 7. Table 1 presents the linear relaxation values for each existing formulation (taken from [22]) and also for the new formulation obtained by applying the DT to the natural formulation.

This small example illustrates the potential power of the DT on the SFP. In fact, *it proves that the weak natural formulation is transformed into a new formulation that can be strictly stronger than any other known SFP formulation in some instances.* On

TABLE 1
Linear relaxation values for the instance in Figure 10.

Formulation	Linear relaxation
Natural (39)	4
Lifted-cut [15]	4
Full directed flow based [17]	6
Tree-based [22]	5
Extended cut-based [22]	5.14
Strengthened extended cut-based [22]	6
Distance transformation	7
Distance transformation $L = 3$	5

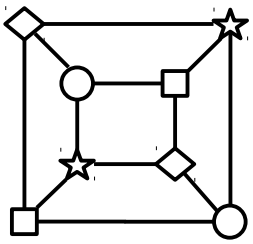


FIG. 10. *SFP instance with four demands (pairs of identical symbols) and unitary edge costs.*

the other hand, the result obtained by setting $L = 3$ shows that the limitation of levels, necessary on larger instances, may affect the linear relaxation bounds significantly.

The following experiments were performed in a single core of a machine with a 2.5 GHz i7 processor and 16 GB of RAM. The tested formulations were implemented via the XPRESS-Optimizer 7.3. We performed tests with three types of instances.

- Small SFP instances (pdh, di-yuan, dfn-gwin, polska and nobel-us) available in the SNDLib.
- Steiner instances c01_st,...,c10_st from the SteinLib. Those instances are defined over random graphs with 500 vertices. For an instance with terminal-set T, we defined D as $\{(r, v) : v \in T,\ v \neq r\}$, where r is the terminal with smaller index. Those instances can be easily solved by Steiner tree problem in graphs (SPG) codes using a directed formulation. Nevertheless, it is interesting to see how the DT can improve the undirected formulation.
- SFP instances c01,...,c10 derived from the above instances as follows. The set D is obtained by pairing consecutive terminals in T. If $|T|$ is odd, an extra demand from the first to the last terminal is included. To the best of our knowledge, some of those instances can be very hard for current solution methods.

Table 2 compares gaps (with respect to optimal or best known UBs) and times to solve linear relaxations for: (1) the natural formulation (39); (2) the lifted-cut formulation [15]; (3) the strengthened extended cut-based (SECB) formulation [22]; (4) a DT reformulation over the natural formulation for $L = 3$, $L = 4$, and $L = 5$, using unitary distances and with a singleton set S containing the first terminal. We state below some comments on these results.

- The lifted-cut formulation gaps are not much better than those from the natural formulation.
- The SECB formulation assumes that demands with a common vertex will be merged into demands containing more than two vertices. The connectivity

TABLE 2
Natural [1], *lifted-cut* [15], *strengthened extended cut-based* [22], *and DT reformulation* ($|S| = 1$) *root gaps.*

					Natural		Lifted-cut		SECB		DT $L = 3$		DT $L = 4$		DT $L = 5$	
Instance	$\|V\|$	$\|E\|$	$\|D\|$	UB	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)
pdh	11	34	27	897551	41.0	0.00	32.2	0.35			0.0	0.01	0.0	0.01	0.0	0.01
di-yuan	11	42	48	21570000	32.8	0.00	26.1	0.01			0.6	0.02	0.0	0.01	0.0	0.01
dfn-gwin	11	47	9	79960	29.8	0.00	28.5	0.00			1.0	0.04	0.9	0.10	0.9	0.11
polska	12	18	17	214100	38.1	0.00	29.2	0.01			0.0	0.01	0.0	0.01	0.0	0.01
nobel-us	14	21	33	8481000	30.5	0.00	20.1	0.01			1.2	0.02	1.2	0.04	0.8	0.16
Avg.					34.4	0.00	27.2	0.08			0.6	0.02	0.3	0.03	0.3	0.06
c01_st	500	625	4	85	16.5	0.03	12.4	0.07			0.0	1.38	0.0	2.05	0.0	3.77
c02_st	500	625	9	144	24.7	0.16	24.7	0.21			0.0	3.37	0.0	3.63	0.0	6.34
c03_st	500	625	82	754	23.1	0.27	20.6	1.27			0.8	2.10	0.8	4.19	0.8	7.26
c04_st	500	625	125	1079	20.8	0.29	19.6	2.40			0.1	1.09	0.1	6.32	0.1	5.99
c05_st	500	625	249	1579	19.5	0.22	18.6	2.07			0.3	1.58	0.3	7.88	0.3	12.29
c06_st	500	1000	4	55	17.3	0.16	14.6	0.28			4.2	4.00	3.6	10.10	4.2	8.89
c07_st	500	1000	9	102	18.6	0.18	14.2	0.44			0.0	1.26	0.0	1.69	0.0	1.79
c08_st	500	1000	82	509	25.5	0.29	24.3	1.27			0.1	4.73	0.1	13.65	0.1	27.45
c09_st	500	1000	124	707	28.3	0.59	26.9	3.40			0.6	14.70	0.6	11.48	0.6	21.30
c10_st	500	1000	249	1093	25.4	0.59	24.4	5.11			0.0	3.22	0.0	5.26	0.0	13.90
Avg.					21.7	0.26	19.3	1.60			1.0	3.92	0.9	7.06	0.9	12.60
c01	500	625	3	85	16.5	0.03	5.3	0.11	0.6	15.6	1.0	3.53	1.0	4.62	1.0	7.34
c02	500	625	5	143	24.1	0.19	18.2	0.31	0.4	23.4	17.5	2.84	17.5	3.03	17.5	3.84
c03	500	625	42	754	23.1	0.39	21.6	0.95	0.1	623.7	18.9	13.80	18.9	21.30	18.9	27.40
c04	500	625	63	1079	20.8	0.57	19.3	1.06	0.8	1800	11.0	23.70	11.0	47.80	11.0	86.70
c05	500	625	125	1579	19.6	4.32	18.7	1.36	8.0	1800	13.9	82.60	13.9	71.00	13.9	134.80
c06	500	1000	3	47	3.2	0.13	3.2	0.38	0.0	1.43	0.0	2.83	0.0	2.91	0.0	4.44
c07	500	1000	5	89	10.1	0.07	8.4	0.32	0.0	19.1	0.0	2.70	0.0	2.51	0.0	2.98
c08	500	1000	42	509	25.5	0.77	25.3	0.92	4.2	1800	18.7	59.40	18.7	111.80	18.7	236.20
c09	500	1000	63	707	28.3	1.20	27.2	2.16	3.9	1800	23.0	39.80	23.0	122.20	23.0	276.60
c10	500	1000	125	1093	25.4	1.38	24.4	2.57	12.6	1800	20.3	113.90	20.3	228.40	20.2	617.90
Avg.					18.5	0.84	16.1	1.00	3.1	968.3	11.4	31.70	11.3	56.40	11.3	127.90
Avg.					22.7	0.35	19.5	1.08			5.2	4.47	5.0	8.31	5.0	14.80

TABLE 3
Root gaps for the dynamic choice of S in DT.

	$L=3$		$L=4$		$L=5$	
Inst.	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)	T (s)
c01	0.98	3.53	0.98	4.62	0.98	7.34
c02	6.29	2.99	5.83	4.73	5.60	8.36
c03	7.36	6.86	7.05	20.84	6.97	28.86
c04	4.50	10.64	4.32	25.12	4.20	31.95
c05	3.17	8.82	3.09	52.4	2.98	37.8
c06	0.00	2.83	0.00	2.91	0.00	4.44
c07	0.00	2.7	0.00	2.51	0.00	2.98
c08	9.89	20.16	9.69	40.14	9.57	62.45
c09	11.33	28.15	10.84	61.33	10.83	98.65
c10	8.12	40.38	7.88	82.15	7.78	135.88
Avg.	4.86	11.96	4.52	27.49	4.45	38.89

requirement of those demands is the existence of paths joining all their vertices. In all SNDLib and C_st instances this preprocessing reduces all demands to a single demand. The SECB formulation for a single demand containing several vertices is exactly equivalent to the directed Steiner formulation. So, it is expected that the gap for those types is nearly zero and the times are small. The demands are not merged on the C instances, so the test of the SECB formulation is more relevant for that type of instance. The experiments show that the gaps obtained are small; however, the number of variables in the formulation ($O(m|D|)$) and the number of times that the min-cut separation algorithm has to be called per separation round ($O(|D|^2)$) make the formulation slow when $|D|$ grows. In five of the C instances the solution was stopped at 1800 seconds, before the cut separation converged.

- The DT can reduce the gaps significantly with respect to the natural formulation. As expected, the average gaps decrease when L increases, but the improvement quickly becomes marginal. The DT with $L=3$ seems to be the best compromise between gap and running times on most instances. While the gap reductions are remarkable for SNDLib and C_st instances, they are less impressive for the C instances. We verified that their distance transformed fractional solutions were divided into a number of connected components. All those components, except the one that contained the vertex in S, are in level L, where vertex splittings do not happen.

In order to make the DT effective on that last case, we devised an iterative scheme for choosing a larger set S. We start with a single vertex in S and solve the linear relaxation of the corresponding DT. While the fractional solution still contains vertices in level L, we introduce one vertex from the larger connected component in L and solve the new DT. Table 3 shows the results of this dynamic procedure for $L=3$, $L=4$, and $L=5$. While the resulting gaps are significantly smaller, they are still larger than those obtained in other types of instances.

Finally, Table 4 compares the results of the full branch-and-cut over the original formulation (39) with the branch-and-cut over the DT reformulation, for some chosen parameterization. We mark in bold the time of the method that could solve the instance faster. If no method could solve the instance, either because the time limit of 7200 seconds was exceeded or because it ran out of memory, we mark in bold the smaller final gap obtained. We did not pass any external upper bound (UB) to the branch-and-cut; those gaps are with respect to the best solution found by the method

itself. Although the harder instances could not be solved to optimality, it is clear that the overall performance of the DT reformulation is much better.

TABLE 4
Comparison of full branch-and-cut over natural formulation and over DT reformulation. Gaps are given with respect to the UBs found by branch-and-cut itself, not to the best known solutions.

Inst.	Nodes	Gap (%)	T (s)	Nodes	Gap (%)	T (s)	
pdh	20183	0.00	10.52	1	0.00	**0.01**	$L = 4$, $\|S\| = 1$
di-yuan	236	0.00	0.19	1	0.00	**0.01**	$L = 4$, $\|S\| = 1$
dfn-gwin	1389	0.00	0.64	5	0.00	**0.10**	$L = 4$, $\|S\| = 1$
polska	332	0.00	0.15	1	0.00	**0.01**	$L = 4$, $\|S\| = 1$
nobel-us	186	0.00	0.09	3	0.00	**0.05**	$L = 4$, $\|S\| = 1$
c01_st	52	0.00	**2.17**	2	0.00	2.65	$L = 3$, $\|S\| = 1$
c02_st	4275	0.00	60.52	1	0.00	**4.56**	$L = 4$, $\|S\| = 1$
c03_st	10887	28.38	o.m.	1212	**0.53**	7200	$L = 7$, $\|S\| = 1$
c04_st	20979	22.89	7200	255	0.00	**6.18**	$L = 7$, $\|S\| = 1$
c05_st	45310	20.52	7200	149	**0.06**	7200	$L = 7$, $\|S\| = 1$
c06_st	241	0.00	**5.48**	14	0.00	10.77	$L = 3$, $\|S\| = 1$
c07_st	11409	0.00	565.24	1	0.00	**1.55**	$L = 3$, $\|S\| = 1$
c08_st	12700	42.44	o.m.	12	0.00	**30.74**	$L = 3$, $\|S\| = 1$
c09_st	16270	43.85	o.m.	514	**0.32**	7200	$L = 3$, $\|S\| = 1$
c10_st	29232	46.02	7200	20	0.00	**292.58**	$L = 4$, $\|S\| = 1$
c01	56	0.00	**2.13**	3	0.00	3.38	$L = 3$, $\|S\| = 1$
c02	4942	0.00	84.54	69	0.00	**41.13**	$L = 4$, $\|S\| = 2$
c03	18113	28.51	7200	119	**10.61**	7200	$L = 7$, $\|S\| = 8$
c04	29877	23.91	o.m.	821	**3.99**	7200	$L = 3$, $\|S\| = 11$
c05	34645	21.47	7200	198	**3.04**	7200	$L = 4$, $\|S\| = 9$
c06	3	0.00	**1.12**	1	0.00	2.39	$L = 3$, $\|S\| = 1$
c07	239	0.00	10.59	2	0.00	**2.33**	$L = 3$, $\|S\| = 1$
c08	12312	41.06	o.m.	439	**8.84**	7200	$L = 4$, $\|S\| = 13$
c09	15243	48.37	o.m.	235	**11.0**	7200	$L = 3$, $\|S\| = 16$
c10	17361	44.28	7200	17	**7.41**	7200	$L = 4$, $\|S\| = 27$

5.2. The survivable network design with hop constraints problem. We consider in this section the HSNDP defined in the introduction. We address the connectivity requirements by using the layered flow formulation described in subsection 4.2, which results in a large extended formulation for the problem. It can be verified that using the flow formulations proposed in subsection 4.2 to model the connectivity requirements of HSNDP results in a formulation that is equivalent to the one originally proposed in [18]. Our objective in this section is to provide numerical evidence that the formulation is well-suited for Benders decomposition. Specifically, we compare the following three approaches for solving the extended formulation. First, we feed the formulation directly into CPLEX, leaving all parameters at their default values. Second, we consider the automatic Benders decomposition algorithm implemented in CPLEX 12.8, which decomposes the extended formulation into a master problem, which contains only the design variables x, w, and y and Benders cuts, and one subproblem for each demand D, which contains the flow variables g associated to that demand together with the flow conservation constraints and capacity constraints described in subsection 4.2. Last, we implement an ad-hoc Benders decomposition algorithm using callbacks and closely following the lines of [3]. Specifically, our algorithm solves the master problem through a branch-and-cut algorithm. Every time an integer solution $(\bar{x}, \bar{w}, \bar{y})$ is found in the branch-and-cut tree, all subproblems are solved to see if the solution $(\bar{x}, \bar{w}, \bar{y})$ is feasible for the original problem. If this is not the case, the subproblems return one or more Benders cuts that are added to the master problem at all nodes of the branch-and-cut tree.

For each algorithm, we feed the solver with the best known solution (denoted BKS in Table 5) and allow it to fathom any node worse than BKS. Table 5 reports the results of our computational experiments. The table also contains the results obtained by using the classical layered formulations from [3] (denoted HOP formulation in the table), where the connectivity requirements are imposed through flows on layered graphs that are built directly on the original graph G. Hence, the HOP formulation contrasts with the HOP-level formulation that models flows on layered graphs that are built on the top of the layered graph $\mathbf{G}$.

Those experiments were carried out on a computer equipped with an Intel® Xeon® processor X5460 3.16 GHz CPU with 32 GB of RAM, using CPLEX 12.8 with Concert Technology for JAVA [13]. The time limit was set to 36000 seconds. The TC and TE instances in our tests are those widely used in the literature. They correspond to complete graphs, vertices are associated to points in the plane, the costs are the Euclidean distances. TC-20 has vertex 0 in the center and demands $D = \{(0, v) : v = 1, \ldots, 20\}$; TE-20 is similar but has vertex 0 in a corner. TC-40 and TE-40 are similar but $|D| = 40$. We performed tests taking $H \in \{3, 4\}$ and $K \in \{2, 3\}$.

Table 5 is divided into two parts. The top part shows results for HOP formulation, and the bottom part the DT-HOP formulation. Columns *root LB* and *root gap* show the lower bounds and gap with respect to the best known solution for HOP and DT-HOP formulations (those values do not depend on whether the Benders decomposition is used or not). The remaining columns are statistics for the exact methods. In particular, the number of nodes is written in thousands (K) or millions (M). Therefore, we are comparing six methods on each instance. We mark in bold the time of the method that could solve the instance the fastest. If no method could solve the instance within the time limit, we mark in bold the best final lower bounds obtained. The following results can be seen.

- DT works very well for this problem. For all instances, a method based on DT-HOP was the winner.
- The Benders decomposition can be a good alternative for mitigating problems related to the large size of the reformulations obtained by DT. DT-HOP with automatic Benders was the best method for all instances, allowing us to close most open instances. In addition, the automatic Benders decomposition performs significantly better than the ad-hoc algorithm.

6. Conclusions. A growing part of integer programming research is devoted to finding new effective extended formulations for certain families of problems. This paper contributes in this direction, introducing a technique with the potential to strengthen existing formulations for a large class of NDPs. The increase in formulation size is not necessarily exaggerated (actually, the increase can be controlled by the parameter L), making the overall approach computationally appealing in a number of cases.

In principle, DT is a generic technique that could be applied on many other network design problems. The question that has to be answered for any candidate problem is whether the reduction of root gaps is significant enough to compensate for the increased size of the formulation. Based on the experiments presented, there are two factors that seem to make the DT more or less suited to a particular NDP.

- The DT seems to work better on NDPs with sparser solutions. For example, on HSNDP instances with $K = 2$ the average gap was reduced by 72%; on HSNDP instances with $K = 3$ a more modest average gap reduction of 43%

TABLE 5

Comparison of HOP and DT-HOP formulations: direct solution by the CPLEX mixed integer program solver through compact or Benders approaches, and ad-hoc branch-and-cut-based Benders decomposition.

						Compact				Benders auto				Benders via callbacks			
		H	K	BKS	Root LB	Root gap (%)	Final LB	Final UB	Nodes	T (s)	Final LB	Final UB	Nodes	T (s)	Final LB	Final UB	Nodes	T (s)
HOP formulation	TC-20	3	2	607	495.1	18.4	607.0	607	67K	2353	607.0	607	3.6M	10748	607.0	607	973K	9294
		3	3	842	769.1	8.7	842.0	842	23K	1068	842.0	841	665K	1835	842.0	842	246K	1253
		4	2	536	442.3	17.5	506.5		57K	36000	508.4		2.8M	36000	523.0		2.4M	36000
		4	3	750	699.7	6.7	750.0	750	16K	8776	750.0	750	421K	1530	750.0	750	286K	2469
	TE-20	3	2	776	603.1	22.3	776.0	776	59K	3508	724.4		6.2M	36000	745.0		1.5M	36000
		3	3	1082	911.6	15.8	1082.0	1082	184K	9733	1011.5		3.8M	36000	1024.9		1.2M	36000
		4	2	670	516.9	22.8	587.5		47K	36000	569.1		1.6M	36000	573.2		720K	36000
		4	3	919	793.0	13.7	847.2		44K	36000	830.4		675K	36000	838.0		558K	36000
	TC-40	3	2	632	516.9	18.2	623.1	632	136K	36000	593.9		3.6M	36000	592.2		472K	36000
		3	3	889	795.0	10.6	881.5	889	127K	36000	856.4		2.9M	36000	853.6		489K	36000
		4	2	538	456.4	15.2	476.4		1.6K	36000	501.9		2.2M	36000	494.0		341K	36000
		4	3	795	710.4	10.6	728.8		1.8K	36000	747.8		2.3M	36000	747.4		437K	36000
	TE-40	3	2	790	616.9	21.9	746.7		37K	36000	704.9		2.1M	36000	695.3		260K	36000
		3	3	1094	932.5	14.8	1045.0		33K	36000	1009.5		2.0M	36000	999.4		319K	36000
		4	2	661	528.4	20.1	546.4		676	36000	568.2		1.5M	36000	558.5		68K	36000
		4	3	968	808.6	16.5	826.9		764	36000	841.5		1.3M	36000	833.7		88K	36000

						Compact				Benders auto				Benders via callbacks			
		H	K	BKS	Root LB	Root gap (%)	Final LB	Final UB	Nodes	T (s)	Final LB	Final UB	Nodes	T (s)	Final LB	Final UB	Nodes	T (s)
DT-HOP formulation	TC-20	3	2	607	581.0	4.3	607.0	607	2.6K	43	607.0	607	3.3K	16	607.0	607	2.8K	191
		3	3	842	798.8	5.1	842.0	842	2.9K	319	842.0	842	11K	60	842.0	842	11K	601
		4	2	536	501.0	6.5	536.0	536	5.2K	20615	536.0	536	15K	148	536.0	536	7.1K	1052
		4	3	750	714.8	4.7	737.2	750	2.5K	36000	750.0	750	28K	392	750.0	750	10K	1872
	TE-20	3	2	776	764.3	1.5	776.0	776	47	27	776.0	776	13	8	776.0	776	9	46
		3	3	1082	1025.5	5.2	1082.0	1082	15K	8955	1082.0	1082	124K	1048	1082.0	1082	182K	12865
		4	2	670	607.0	9.4	623.4		834	36000	670.0	670	239K	5454	652.3		128K	36000
		4	3	919	843.2	8.3	854.4		692	36000	886.5		881K	36000	876.7		98K	36000
	TC-40	3	2	632	595.8	5.7	632.0	632	2.0K	317	632.0	632	11K	183	632.0	632	2.4	2169
		3	3	889	840.1	5.5	889.0	889	12K	11099	889.0	889	141K	4361	882.0		46K	36000
		4	2	538	515.8	4.1	527.5		189	36000	538.0	538	868	258	538.0	538	817	10349
		4	3	795	738.6	7.1	743.1		75	36000	780.1	**785**	297K	36000	763.6	**785**	10K	36000
	TE-40	3	2	790	758.3	4.0	790.0	790	789	688	790.0	790	5.7K	202	790.0	790	2.1K	2239
		3	3	1094	1025.2	6.3	1062.5		4.18K	36000	1094.0	1094	456K	21341	1065.3		31K	36000
		4	2	661	614.4	7.1	618.2		25	36000	654.0	**654**	33K	5585	623.6		1.1K	36000
		4	3	968	857.4	11.4	859.9		23	36000	883.7		126K	36000	860.5		57	36000

was obtained. This is consistent with the theory presented in section 2, which asserts that DT works by splitting nodes according to the different distances (to the sources) induced by decomposing fractional solutions. Denser fractional solutions provide more ways of performing the decomposition, resulting in fewer node splittings.

- The DT seems to work better on problems where the solutions are likely to have a "small diameter." For example, this happens on HSNDP because the of the hop constraint. This allows small values of L to be used without compromising the strength of the reformulation. Recall that SFP solutions are very sparse, but typically have large diameter. Although the latter characteristic is not favorable, the DT with limited values of L still provided good results.

There are also important issues that need to be addressed in future research on DT. One of them is devising a general method for choosing the set S and the limit L. The experiments on SFP with ad-hoc methods showed that the choice of these values can make a lot of difference.

Acknowledgment. The authors would like to thank the referees for suggesting numerous improvements to the manuscript.

REFERENCES

[1] Y. P. Aneja, *An integer linear programming approach to the Steiner problem in graphs*, Networks, 10 (1980), pp. 167–178.

[2] F. Bendali, I. Diarrassouba, A. Mahjoub, and J. Mailfert, *The k edge-disjoint 3-hop-constrained paths polytope*, Discrete Optim., 7 (2010), pp. 222–233.

[3] Q. Botton, B. Fortz, L. Gouveia, and M. Poss, *Benders decomposition for the hop-constrained survivable network design problem*, INFORMS J. Comput., 25 (2013), pp. 13–26.

[4] S. Chopra and M. R. Rao, *The Steiner tree problem* I: *Formulations, compositions and extension of facets*, Math. Program., 64 (1994), pp. 209–229.

[5] S. Chopra and M. R. Rao, *The Steiner tree problem* II: *Properties and classes of facets*, Math. Program., 64 (1994), pp. 231–246.

[6] M. X. Goemans and D. P. Williamson, *The primal-dual method for approximation algorithms and its application to network design problems*, PWS, Boston, 1997, pp. 144–191.

[7] L. Gouveia, *Multicommodity flow models for spanning trees with hop constraints*, European J. Oper. Res., 95 (1996), pp. 178–190, https://doi.org/10.1016/0377-2217(95)00090-9.

[8] L. Gouveia, M. Leitner, and I. Ljubić, *Hop constrained Steiner trees with multiple root nodes*, European J. Oper. Res., 236 (2014), pp. 100–112.

[9] L. Gouveia, M. Leitner, and I. Ljubić, *The two-level diameter constrained spanning tree problem*, Math. Program., 150 (2015), pp. 49–78.

[10] L. Gouveia, L. Simonetti, and E. Uchoa, *Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs*, Math. Program., 128 (2011), pp. 123–148.

[11] D. Huygens and A. Mahjoub, *Integer programming formulations for the two 4-hop-constrained paths problem*, Networks, 49 (2007), pp. 135–144.

[12] D. Huygens, A. Mahjoub, and P. Pesneau, *Two edge-disjoint hop-constrained paths and polyhedra*, SIAM J. Discrete Math., 18 (2004), pp. 287–312.

[13] IBM, *IBM ILOG CPLEX V*12.1 : *Users' Manual for CPLEX*, International Business Machines Corporation, New York, 2009, p. 157.

[14] T. Koch, A. Martin, and S. Voss, *SteinLib: An updated library on Steiner tree problems in graphs*, in Steiner Trees in Industry, X. Z. Cheng and D. Z. Du, eds., Comb. Optim. 11, Springer, Boston, MA, 2001, pp. 285–325.

[15] J. Könemann, S. Leonardi, G. Schäfer, and S. H. van Zwam, *A group-strategyproof cost sharing mechanism for the Steiner forest game*, SIAM J. Comput., 37 (2008), pp. 1319–1341.

[16] I. Ljubić and S. Gollowitzer, *Layered graph approaches to the hop constrained connected*

facility location problem, INFORMS J. Comput., 25 (2013), pp. 256–270.
[17] T. L. Magnanti and S. Raghavan, *Strong formulations for network design problems with connectivity requirements*, Networks, 45 (2005), pp. 61–79.
[18] A. R. Mahjoub, L. Simonetti, and E. Uchoa, *Hop-level flow formulation for the survivable network design with hop constraints problem*, Networks, 61 (2013), pp. 171–179.
[19] M. Poggi de Aragão, E. Uchoa, and R. Werneck, *Dual heuristics on the exact solution of large Steiner problems*, Electron. Notes Discrete Math., 7 (2001), pp. 150–153.
[20] T. Polzin and S. Daneshmand, *Improved algorithms for the Steiner problem in networks*, Discrete Appl. Math., 112 (2001), pp. 263–300.
[21] M. Ruthmair and G. R. Raidl, *A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems*, in Integer Programming and Combinatoral Optimization, IPCO 2011, O. Günlük and G. J. Woeginger, eds., Lecture Notes in Comput. Sci. 6655, Springer, Berlin, 2011, pp. 376–388.
[22] D. Schmidt, B. Zey, and F. Margot, *MIP Formulations for the Steiner Forest Problem*, preprint, https://arxiv.org/abs/1709.01124, 2017.
[23] R. Wong, *A dual ascent approach for Steiner tree problems on a directed graph*, Math. Program., 28 (1984), pp. 271–287.