Extended formulation for hop constrained distribution network configuration problems

J. De Boeck[a], B. Fortz[a]

[a]*Département d'informatique, Université Libre de Bruxelles, CP 212, Boulevard du Triomphe, 1050 Bruxelles, Belgium and INOCS, INRIA Lille Nord-Europe, France*

Abstract

A distribution network is a system aiming to transfer a certain type of resource from feeders to customers. Feeders are producers of a resource and customers have a certain demand in this resource that must be satisfied. Distribution networks can be represented on graphs and be subject to constraints that limit the number of intermediate nodes between some elements of the network (hop constraints) because of physical constraints. This paper uses layered graphs for hop constrained problems to build extended formulations. Preprocessing techniques are also presented to reduce the size of the layered graphs used.

The presented model is studied on the hop-constrained minimum margin problem in an electricity network. This problem consists of designing a connected electricity distribution network, and to assign customers to electricity feeders at a maximum number of hops H so as to maximize the minimum capacity margin over the feeders to avoid an overload for any feeder.

Numerical results of our model are compared with those of state-of-the-art solution techniques of the minimum margin problem form Rossi et al. [20]. Variations of the initial problem are also presented, considering losses due to transportation or by replacing hop constraints by distance constraints, a variation arising in the context of multicast transmission in telecommunications.

Keywords: Distribution networks, Hop constraints, Extended formulations, Layered graphs

Highlights:

- Study of hop constrained configuration problems in distribution networks
- Extended formulations using layered graphs for hop constrained problems
- Preprocessing on layered graphs
- Experimental assessment of the interest of preprocessing techniques

Email addresses: jdeboeck@ulb.ac.be (J. De Boeck), bfortz@ulb.ac.be (B. Fortz)

1. Introduction

Large scale distribution network problems appeared in the 19^{th} century with the development of industry, railway, telecommunications and electricity. These distribution network problems have been getting larger and more complex since their introduction. Consider electricity distribution networks. The demand in electricity has been constantly increasing since 1950 with a larger number of customers [16] and are getting closer to their limits [5]. Failure in electricity systems have growing negative impacts, and distribution must become increasingly more reliable [19]. The energy production technologies are also evolving with the apparition of new production systems (solar panels, wind turbines, ...) meaning a larger number of energy producers to consider [19]. In telecommunications distribution networks, the evolution is similar to electricity. Over the past years, data exchange has even been increasing at a much higher rate than electricity demands [22], and reliability is also an important issue.

Electricity and telecommunications *Distribution Network Configuration Problems* (DNCP) have similar constraints. In both problems, a set of feeders is given. Each feeder produces a specific resource with limited capacity. A set of customers must be assigned to the feeders through an existing network in order to satisfy the demands of the customers. Each customer is assigned to a single feeder and the customers assigned to a feeder must form a connected component, potentially using some Steiner nodes. A Steiner node is as a node with a demand equal to zero that does not necessarily need to be assigned to a feeder but could be assigned to ensure connectivity. For electricity distribution networks, feeders are power generators, customers have an electricity demand and connectivity is needed to ensure the electricity transfer from feeders to customers. In the context of telecommunications, distribution networks appear in multicast routing problems [17]. The feeders are data providers and the customers are data relays with a demand in data. Feeders can send data to customers in a certain range and customers can repeat the signal to transfer data to other customers. Connectivity is needed to ensure that a customer can receive the desired informations from its feeder or from a repetitor assigned to the same feeder.

Many DNCP include hop constraints that limit the maximum number of edges between a customer and its feeder to a given value H (HDNCP). Hop constraints are used to model reliability issues. Consider a customer c assigned to its feeder through a path P of length d. If there is a failure on a vertex or an edge of P, customer c would needed to be supplied through another path. The bigger the value of d, the higher the risk of failure in distribution [14]. The value of H can be fixed to limit the maximum probability of failure of transmission to customers. Furthermore, in telecommunications networks, there are data transfer delays due to distances between the transmitter and the receiver. Hop constraints can be used to limit these delays.

Hop constraints were already considered in several hop constrained network design problems by Balakrishnan and Altinkemer [1] and Pirkul and Soni [18] to mention some of them. These problems have often been studied using extended formulations with variables keeping track of the distances of vertices and/or edges from a root in a solution. Over time, layered graphs, introduced by Gouveia [11], have been used for problems including hop constraints to build extended formulations. A layered graph derived from a graph G is a graph containing $H + 1$ layers, layer 0 containing a root r and layer h containing vertices of G that can be reached through a path of length h from r. Layered graphs have been used in hop constrained Steiner tree problems [12, 13, 21], facility location problems [14], survivable network design [2].

In this paper, we study a Mixed Integer Programming (MIP) formulation of HDNCP using layered graphs and study a particular case, the *Minimum Margin Problem* (MMP), introduced by Rossi et al. [20]. This problem finds a feasible solution to HDNCP trying to prevent an overload for

feeders in the case of a sudden increase in demand from some customers. We present an extended formulation that illustrates how extended formulations can reduce the size and strengthen the initial model [4]. Layered graphs have often been used in the previously mentioned problems to find a valid set of edges composing a feasible solution. In HDNCP, layered graphs are used in a different way to find valid assignments of customers to feeders, not taking into consideration which edges are used. A set of reductions of layered graphs is also studied to give a tighter extended formulation.

As a first variant of HDNCP, we study the case where there is a loss of resources due to transportation, as in electricity network [20]. The extended formulation on layered graphs allows to integrate a loss function. This loss function gives for each customer the amount of extra resource needed to compensate losses due to the number of hops in a solution. A second variant of HDNCP is adapted for distance constrained problems where edges have a length and a path between each customer and their feeder can not exceed a certain maximum length D. This occurs in telecommunications network such as in multicast routing problems to avoid delays in data transmission [17].

This paper is organized as follows. The *Minimum Margin Problem* is studied in Section 2. The problem definition and the formulations of Rossi et al. are given in Sections 2.1 and 2.2. A layered formulation using layered graph and preprocessing techniques are presented in Section 2.3. Different formulations as well as the impact of parameters of MMP are analyzed in Section 2.4. Variants of MMP are also studied, integrating resource losses in Section 3, and replacing the hop constraints by distance constraints in Section 4. Conclusions are made in Section 5.

2. Minimum margin problem

2.1. Problem definition

A distribution network is a system aiming to transfer a certain type of resource from feeders to customers. Feeders are producers of a resource and customers have a certain demand in this resource that must be satisfied. We consider in this paper distribution networks with a single resource. The *Hop constrained Distribution Network Configuration Problem* (HDNCP) consists of assigning customers to feeders in a distribution network in order to build connected components respecting the *hop constraints* for each feeder. Hop constraints limit the length of the shortest paths between each feeder and the customers assigned to it to a maximum value H. The network is represented by a graph $G = (V, E)$, $V = V_t \cup V_f$ where V_t is the set of terminal nodes and V_f is the set of nodes representing feeders, $V_f \cap V_t = \emptyset$. In the following $|V_t| = n$ and $|V_f| = m$. The set V_t of terminal nodes is partitioned into a set of customers V_c with positive demand that need to be connected to a feeder and a set of Steiner nodes V_s ($V_t = V_c \cup V_s, V_s \cap V_c = \emptyset$). Consider a set $S = \{S_j\}_{j \in V_f}$ of disjoint subsets of nodes $S_j \subseteq V$. Set S is a feasible solution of HDNCP if:

- each set S_j contains feeder j and each terminal node $i \in V_t$ assigned to j,
- all customers $i \in V_c$ are assigned to exactly one set S_j,
- for each feeder $j \in V_f$, component $C_j^S = (S_j, E(S_j))$, with $E(S_j) = \{uv \in E | u, v \in S_j\}$, is connected and respects the hop contraints.

Consider a set of subtrees $\{T_j\}_{j \in V_f}$ of G, each one rooted in j. If S_j represents the vertices of tree T_j and $S = \{S_j\}_{j \in V_f}$ is a feasible solution, then T_j is a spanning tree in component C_j^S and satisfies the hop constraints. A same solution S can be built from different sets of subtrees $\{T_j\}_{j \in V_f}$ of G as each component C_j^S can have multiple spanning trees respecting hop constraints.

We study a particular case of HDNCP, the *Minimum Margin Problem* (MMP). Customers $i \in V_c$ have a demand $dem_i > 0$, while Steiner nodes of V_s have no demand ($dem_i = 0$ for all $i \in V_s$). Each feeders $j \in V_f$ has a capacity c_j, and its margin is defined as $M_j = c_j - \sum_{i \in S_j \setminus j} dem_i$. The MMP consists of finding a feasible solution of HDNCP maximizing the minimum margin of the feeders $M_{min} = \min\{M_j | j \in V_f\}$ in order to prevent an overload for feeders if the demand of a group of customers increases. Rossi et al. [20] showed that MMP is strongly *NP-hard* by reduction from 3-Partition [9].

We denote by d_{ij} the minimum distance in G between feeder $j \in V_f$ and terminal nodes $i \in V_t$ without going through any other feeder, where the distance between two vertices is defined as the minimum number of edges in a path between these vertices. If no such path exists, $d_{ij} = +\infty$. Set $N_j = \{i \in V_t | d_{ij} \leq H\}$ is the set of terminal nodes that can potentially be assigned to j. The minimum feasible distance d_{min} is the minimum distance for which each customer in V_c can be assigned to at least one feeder.

2.2. Vertex and edge formulations

Rossi et al. [20] proposed two MIP formulations to solve MMP for feeders having equal capacity and using no Steiner nodes, that is, $c_j = c_{j'}$ for all $j, j' \in V_f, j \neq j'$ and $V_s = \emptyset$. These formulations could easily be adapted to the general case considering various capacities and Steiner nodes. The first is an *Edge Formulation* (EF) finding components C_j^S maximizing M_{min} through a network flow problem. If G is connected, EF has $O(|E|)$ variables and $O(|V|^2)$ constraints. Next to the compact number of constraints of EF, one motivation of this formulation was the efficiency of commercial solvers when solving network flow problem and some of its variations [23]. The second formulation proposed is a *Vertex Formulation* (VF) using assignment binary variables x_{ij}.

$$x_{ij} = \begin{cases} 1 & \text{if } i \in V_t \text{ is assigned to } j \in V_f, \\ 0 & \text{otherwise.} \end{cases}$$

Based on values of variables x_{ij}, sets $\{S_j\}_{j \in V_f}$ are defined by $S_j = \{j\} \cup \{i \in V_t | x_{ij} = 1\}$. The VF formulation of Rossi et al. [20] is:

$$\text{Max} \quad M_{min} \tag{1}$$

$$\text{s.t.} \quad c_j - \sum_{i \in N_j} x_{ij} dem_i \geq M_{min} \qquad \forall j \in V_f \tag{2}$$

$$\sum_{j \in V_f} x_{ij} = 1 \qquad \forall i \in V_t \tag{3}$$

$$x_{ij} = 0 \qquad \forall j \in V_f, i \notin N_j \tag{4}$$

$$x_{ij} \leq \sum_{k \in P_{ij}} x_{kj} \qquad \forall j \in V_f, d \in \{2, \ldots, H\}, i \in L_{j,d}^s \tag{5}$$

$$C_j^S \text{ is connected} \qquad \forall j \in V_f \tag{6}$$

$$\text{Distance between terminal node } i \text{ and its assigned feeder } j \text{ is at most } H \text{ in } C_j^S \qquad \forall i \in V_t \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V_t, j \in V_f \tag{8}$$

In this formulation, P_{ij} is the set of terminal nodes at distance $d_{ij} - 1$ of j in G. Set $L^s_{jd} = \{i \in V_c | d_{ij} = d\}$, is the set of terminal nodes i at shortest distance d from j in G. This formulation contains $O(mn)$ variables, which is less than EF. The drawback is the number of constraints needed to ensure the connectivity of the components (6) and the hop constraints (7). These constraints appear in exponential number in VF. Rossi et al. proposed a cutting plane algorithm, starting only with margin constraints (2), assignment constraints (3), constraints (4) for terminal nodes that cannot be assigned to a feeder and *layered constraints* (5). These layered constraints represent the fact that if a terminal node i is at minimum distance $d > 1$ of a feeder j in G, i can be assigned to j only if another terminal node at minimum distance $d-1$ is also assigned to j. Layered constraints are necessary for connectivity of solutions but not sufficient. The cutting plane algorithm adds progressively connectivity (6) and hop constraints (7) of formulation VF.

Complete descriptions of formulations EF and VF can be found in [20]. Numerical results show significantly better computation times for VF.

2.3. Layered Formulation

In the following we present an extended formulation for MMP, then we propose some preprocessing techniques to reduce the size of the formulation.

2.3.1. Layered Graphs

In order to solve the Hop Constrained Minimal Spanning/Steiner Tree problem on a graph G, Gouveia [11] introduced the *expanded acyclic graph* associated to G to derive an extended formulation for these problems. These extended formulations keep track of the distance between an edge and the root in a solution. This expanded graph representation has been used in several hop constrained problem [2, 12, 13, 14] and has often been renamed as *layered graph*. A layered graph associated to a graph $G = (V, E)$, rooted in $v \in V$, with hop limit H is a directed graph $G^H_v = (V^H_v, A^H_v)$ containing all paths of length at most H rooted in v in G. Paths considered contain v only once as their root. Graph G^H_v contains a vertex u_d for each vertex $u \in V$ such that there exists a path of length d from v to u in G, for $1 \leq d \leq H$. A *layer* L_{vd} is the set of vertices u_d for a fixed $d \geq 1$, and there is an extra layer L_{v0} containing only the root v. Set V^H_v is the set of all vertices of the layered graph:

$$V^H_v = \{v\} \cup \{u_d | u \in V, 1 \leq d \leq H, u \in L_{vd}\}$$

and

$$A^H_v = \{vu_1 | vu \in E\} \cup \{u_d u'_{d+1} | u_d, u'_{d+1} \in V^H_v \setminus /\{v\},\ uu' \in E\}$$

is the corresponding set of arcs.

The construction of layered graph G^H_v can be done in $O(n^2 H)$ as follows:

- Set $V^H_v := \{v\}$, $L_{v0} := \{v\}$.
- For $d := 1$ to $H - 1$:
  For each u in L_{vd}:
  For each neighbour $u' \neq v$ of u:
  Set $L_{v\,d+1} := L_{v\,d+1} \cup \{u'_{d+1}\}$, $V^H_v := V^H_v \cup \{u'\}$ u'_{d+1} and $A^H_v := A^H_v \cup \{u_d u'_{d+1}\}$.

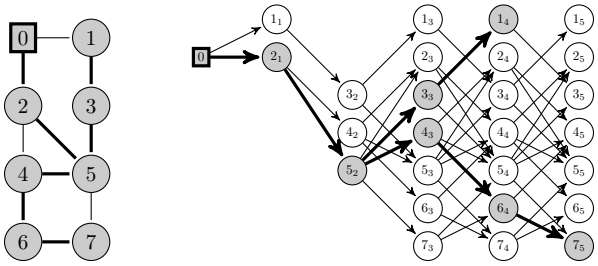

Figure 1: Graph G with spanning tree T_0 with its representation T_0^5 in G_0^5

A layered graph contains at most $O(nH)$ nodes and $O(n^2H)$ arcs. We define the sets $P_{uvd} = \{u' \in L_{vd-1} | u'_{d-1}u_d \in A_v^H\}$, with $1 \leq d \leq H$, as the sets of terminal nodes that are predecessors of u_d in G_v^H. These sets can be built during construction of G_v^H.

A tree $T_v \subseteq G$ rooted in $v \in V$ can be represented as a tree T_v^H in G_v^H if and only if the maximum distance between v and all leaves of T_v is at most H, otherwise T_v^H would need vertices beyond layer H. An illustration of a spanning tree on a layered graph is given in Figure 1.

In numerical results presented in the following Section 2.4, a set of bipartite graphs will be used. Layered graphs derived from bipartite graphs can have a lower number of vertices than when using random graphs.

Lemma 1. *If $G = (V, E)$ is a bipartite graph, then for any $u, v \in V, u \neq v$, u cannot appear in two consecutive layers of G_v^H.*

Proof. In a bipartite graph $G = (V, E), V = V_1 \cup V_2$, paths between vertices in the same set V_i are of even length and paths between vertices in different sets V_i are of odd length. Suppose vertex u appears in two consecutive layers d and $d+1$, that is there exists a path of length d and another of length $d+1$ from v to u in G, this contradicts the bipartite hypothesis. □

2.3.2. Extended formulation of MMP solutions using layered graphs

Feasible solutions of the MMP can be represented on a set of m layered graphs $G_j^H = (V_j^H, A_j^H)$, one for each feeder $j \in V_f$. The layered graph G_j^H is built from G without using feeders different than j, that is, a feeder $j' \neq j$ will not appear in any layer L_{jd}. The set $G^H = \bigcup_{j \in V_f} G_j^H$ is the set of all layered graphs.

Lemma 2. *A feasible solution S of MMP can be represented on G^H if and only if $H \geq d_{min}$*

Proof. Consider a feasible solution S of MMP and all its connected components C_j^S on G. For all component C_j^S, consider a shortest path tree T_j^S. As C_j^S respects the hop constraints, T_j^S respects them as well. Tree T_j^S is representable on G_j^H if and only if the maximum distance between j and its assigned customers is at most H. By definition, d_{min} is the minimum distance for which all customers can be assigned to a feeder, so $d_{min} \leq H$. □

Lemma 2 gives a lower bound for H, so we consider in the following $d_{min} \leq H$.

Figure 2 gives an example of a feasible solution of MMP on G^5. Square nodes 0 and 1 are two feeders and circle nodes 2 to 15 are terminal nodes. The color of each customer is the same as the

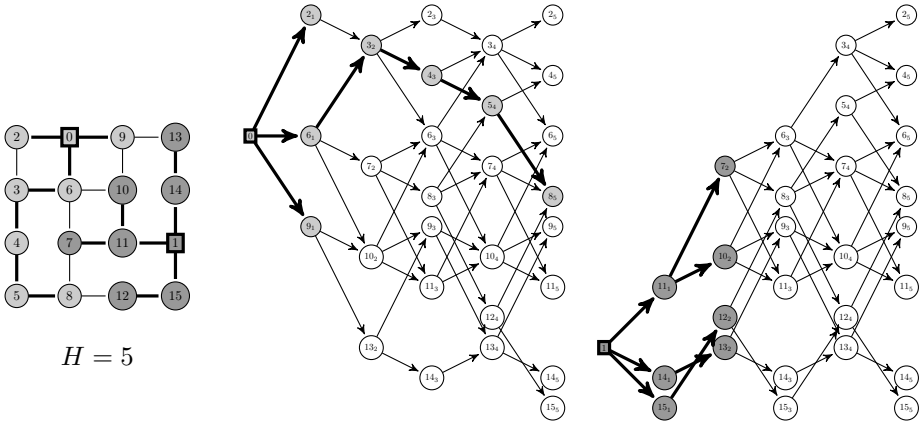

Figure 2: Solution of MMP of a network G on G^5

color of the feeder it is assigned to. The graph used in Figure 2 is bipartite, each terminal node does not appear in two consecutive layers of G_0^5 or G_1^5 as stated in Lemma 1.

An extended compact formulation of MMP is obtained using G^H and assignment-distance binary variables x_{ijd}:

$$x_{ijd} = \begin{cases} 1 & \text{if } i \in V_t \text{ is assigned to } j \in V_f \text{ at a distance } 1 \leq d \leq H, \\ 0 & \text{otherwise.} \end{cases}$$

Variables x_{ijd} define the value of x_{ij} through $x_{ij} = \sum_{1 \leq d \leq H} x_{ijd}$. Assignments of terminal nodes at a distance satisfying the hop constraints are ensured by this relation and (3), which allows to remove hop constraints (7) from VF. Sets S_j are defined as in VF, $S_j = \{j\} \cup \{i \in V_t | x_{ij} = 1\}$. In a feasible solution S of HDNCP represented on G^H, each terminal node $i \in V_t$ is assigned to a single layered graph G_j^H in exactly one layer d if $i \in V_c$ or in at most one layer if $i \in V_s$. Moreover if a terminal node $i \in V_t$ is assigned to j at distance d, that is $x_{ijd} = 1$, then $i \in L_{jd}$ has at least one predecessor assigned to the same feeder in layer L_{jd-1} (if $d > 1$), otherwise C_j^S cannot be connected. Based on this observation, the exponential number of connectivity constraints (6) that is required in VF can be replaced by a compact number of connectivity constraints based on variables x_{ijd} and predecessor sets $P_{ijd} = \{k \in L_{jd-1} | k_{d-1} i_d \in A_j^H\}$:

$$x_{ijd} \leq \sum_{k \in P_{ijd}} x_{kjd-1} \quad \forall j \in V_f, d \in \{2, \ldots, H\}, i \in L_{jd}$$

These connectivity constraints ensure connectivity of components C_j^S for all feeders. The number of these constraints linearly depends on n, m and H.

This leads to the *Layered Formulation* (LF) of MMP:

$$\text{Max } M_{min} \tag{9}$$

$$\text{s.t. } c_j - \sum_{i \in N_j} \sum_{1 \leq d \leq H} x_{ijd} dem_i \geq M_{min} \qquad \forall j \in V_f \tag{10}$$

$$\sum_{j \in V_f} \sum_{1 \leq d \leq H} x_{ijd} = 1 \qquad \forall i \in V_c \tag{11}$$

$$\sum_{j \in V_f} \sum_{1 \leq d \leq H} x_{ijd} \leq 1 \qquad \forall i \in V_s \tag{12}$$

$$x_{ijd} \leq \sum_{k \in P_{ijd}} x_{kj(d-1)} \qquad \forall j \in V_f, d \in \{2, \ldots, H\}, i \in L_{jd} \tag{13}$$

$$x_{ijd} = 0 \qquad \forall j \in V_f, d \in \{1, \ldots, H\}, i \notin L_{jd} \tag{14}$$

$$x_{ijd} \in \{0, 1\} \qquad \forall j \in V_f, i \in V_t, d \in \{1, \ldots, H\} \tag{15}$$

Constraints (10) bound M_{min} with the feeders' margins, constraints (11) and (12) respectively ensure customers are assigned to a single feeder at distance less or equal to H and Steiner nodes are assigned to at most one feeder. Constraints (13) ensure connectivity and constraints (14) set variables x_{ijd} to 0 when a terminal node i cannot be assigned at distance d to feeder j, that is, if $i \notin L_{jd}$. Variables set to 0 are removed from the formulation during preprocessing but we keep them here to allow comparaisons of the solution space of LF with further formulations.

Lemma 3. *Formulation LF is valid for MMP.*

Proof. Consider a feasible solution (x_{ijd}) of LF and the associated partition of nodes $S = \{S_j\}_{j \in V_f}$ of MMP where $S_j = \{i \in V_t : \sum_{1 \leq h \leq H} x_{ijd} = 1\}$. We need to show that each customer $i \in V_c$ is assigned and connected to its feeder at maximum distance H through S. Constraints (11) ensure that each customer i in V_c is assigned to exactly one feeder j at a certain distance d with $x_{ijd} > 0$. If d=1, i is a neighbour of j and connectivity is ensured. If $1 < d \leq H$, connectivity constraints (13) ensure there exists a variable $x_{i'jd-1} > 0$, with $i' \neq i$ and $ii' \in E$ so i is connected to i' in C_j^S. As $i' \in P_{ijd}$, i' is closer to j than i. Constraints (13) can be used to find a path connecting i to j in C_j^S. This procedure will terminate after at most $H - 1$ iterations as d is decreased at each iteration, leading to paths of maximum length H. □

Constraints (11)-(15) guarantee connectivity of solutions as well as respect of hop constraints. The number of variables and constraints is $O(mnH)$. The value of H influences the size of LF. If H has the minimum value d_{min}, some customers can be assigned to a single feeder at a single distance, fixing some assignments during preprocessing. Increasing H might increase the number of feeders a customer could be assigned to leading to a combinatorially more complex problem and a possible better optimal value. We analyse the influence of H on the computation time and the optimal value in Section 2.4.4.

The number of binary variables in LF depends on H and can be considerably larger than in VF. To reduce the number of such variables, variables x_{ij} are reintroduced and a relaxation is performed

over variables x_{ijd} in the *Layered Formulation Relaxation* (LFR):

$$\text{Max } M_{min} \tag{16}$$

$$\text{s.t. } c_j - \sum_{i \in N_j} x_{ij} dem_i \geq M_{min} \qquad \forall j \in V_f \tag{17}$$

$$\sum_{j \in V_f} x_{ij} = 1 \qquad \forall i \in V_c \tag{18}$$

$$\sum_{j \in V_f} x_{ij} \leq 1 \qquad \forall i \in V_s \tag{19}$$

$$x_{ij} = \sum_{1 \leq d \leq H} x_{ijd} \qquad \forall j \in V_f, i \in N_j \tag{20}$$

$$x_{ijd} \leq \sum_{k \in P_{ijd}} x_{kj(d-1)} \qquad \forall j \in V_f, d \in \{2, \ldots, H\}, i \in L_{jd} \tag{21}$$

$$x_{ijd} = 0 \qquad \forall j \in V_f, d \in \{1, \ldots, H\}, i \notin L_{jd} \tag{22}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in V_f, i \in N_j \tag{23}$$

$$0 \leq x_{ijd} \leq 1 \qquad \forall j \in V_f, d \in \{1, \ldots, H\}, i \in L_{jd} \tag{24}$$

Lemma 4. *Formulation LFR is valid for MMP.*

Proof. The result follows from the fact that there always exist a feasible solution of LFR where x_{ijd} are binary if x_{ij} are binary. Indeed, given a solution (x_{ij}, x_{ijd}) of LFR with x_{ijd} fractional, a integer solution (x_{ij}, x'_{ijd}) with the same cost can be derived by setting $x'_{ijd} = 1$ for the smaller d such that $x_{ijd} > 0$ and $x_{ijd'} = 0$ for $d' \neq d$. It is then easy to see that x'_{ijd} is a solution of LF, hence of MMP. □

Let $\mathcal{S}^F$ be the solution space of a formulation F and $\overline{F}$ be the LP-relaxation of F. When substituting variables x_{ij} by $\sum_{1 \leq d \leq H} x_{ijd}$ in the LP-relaxation of LFR using constraint (20), we get the LP-relaxation of LF, leading to $proj_{x_{ijd}}(\mathcal{S}^{\overline{\text{LF}}}) = proj_{x_{ijd}}(\mathcal{S}^{\overline{\text{LFR}}})$. The number of binary variables in LFR is the same than in VF and at most the same than in LF as each variable x_{ij} in LFR corresponds to at least one variable x_{ijd} in LF. Computational results using LF and LFR will be made using state-of-the-art MIP solvers in Section 2.4.3.

2.3.3. Preprocessing of layered graphs

A feasible solution $S = \{S_j\}_{j \in V_f}$ of MMP defines a set of connected components C_j^S in a graph $G = (V, E)$. All components can be rebuilt from a spanning tree $T_j^S \subseteq C_j^S$ satisfying hop constraints, in particular from any shortest path tree of C_j^S. When searching in G_v^H for the representation of a shortest path tree $T_v \subseteq G$ of a subset of vertices $V' \subseteq V$, some reductions can be applied to G_v^H to remove some arcs and vertices that cannot appear in a shortest path tree. Let $\delta^-(u_d)$ be the number of incoming arcs of u_d in G_v^H and $G(V') = (V', E(V'))$ be the induced subgraph of G for a set of vertices $V' \subseteq V$, $E(V') = \{uv \in E | u, v \in V'\}$.

We define a *redundant* vertex/arc of G_v^H as a vertex or arc that cannot appear in the representation on G_v^H of a shortest path rooted in v in G', for any induced subgraph $G' = G(V')$ where $V' \subseteq V$ contains v.

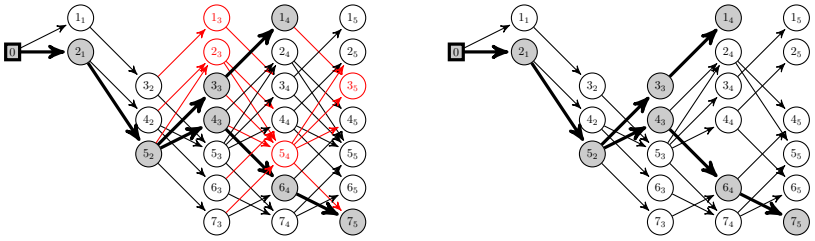

Figure 3: Simple path reductions

Redundant vertices and arcs can be removed from G_v^H as they will not appear in the representation on G_v^H of any shortest path tree of a subgraph $G(V'), V' \subseteq V$ containing v. All vertices and arcs that are accessible in G_v^H only through a redundant vertex or arc can also be eliminated from G_v^H. We introduce in the following a series of reductions that eliminate redundant vertices and arcs. These reductions eliminate form G^H the possibility of representing some spanning trees of components C_j^S that are not shortest path trees. In the following figures used to illustrate reductions techniques, red vertices and arcs in the left-hand side graph are those eliminated from the original layered graph by a specific reduction technique, and the right-hand side graph contains the modified tree representation of the tree in Figure 1 with possibly shorter paths.

Simple path reductions

In a tree T_v of G rooted in v, all paths between v and a vertex of T_v are simple paths, that is, paths where each vertex of G appears at most once. In G_v^H, if a vertex u_d has only one incoming arc from a vertex u'_{d-1} , the outgoing arc going to u'_{d+1} going back to u' will never be used in a shortest path tree T_v rooted in v. *Simple Path Reductions* (SPR) remove such arcs from G_v^H, as well as vertices that have no more incoming arc after eliminating these arcs.

Lemma 5. *Let u be a node of in-degree 1 at level d (i.e. $\delta^-(u_d) = 1$) and let u'_{d-1} be the unique predecessor of u_d in G_v^H. Then arc $u_d u'_{d+1}$ is redundant.*

Proof. Assume there exists a node u such that $\delta^-(u_d) = 1$, $u'_{d-1}u_d \in A_v^H$ and $u_d u'_{d+1} \in A_v^H$ is not redundant. Then there exists a subgraph $G' = G(V')$, where $V' \subseteq V$ contains v, and a shortest path P from v to u' in G' rooted in v such that arc $u_d u'_{d+1}$ belongs to P. Since $u'_{d-1}u_d$ is the only arc incoming in u_d, it also belongs to P. But then the subpath of P from v to u'_{d-1} is shorter than P, contradicting the fact that P is a shortest path. Hence $u_d u'_{d+1}$ is redundant.

□

Figure 3 gives the layered graph G_v^H with SPR of the graph considered in Figure 1. The SPR can be applied to a layered graph G_v^H as follows:

- For $d := 2$ to $H - 1$:
  For each vertex $u \in L_{vd}$:
  Check the number $|P_{uvd}|$ of incoming arcs of u_d,
  If $|P_{uvd}| = 1$, $P_{uvd} = \{u'\}$ and $u_d u'_{d+1} \in A_v^H$:
  Remove $u_d u'_{d+1}$ and remove u from predecessors $P_{u'vd+1}$.
  If $|P_{uvd}| = 0$:

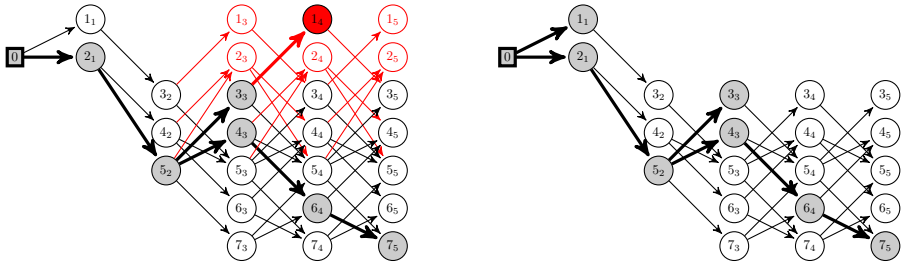

Figure 4: Root neighbour reductions

Remove u_d of V_v^H as well as all outgoing arcs $u_d u'_{d+1}$,
Remove u from all predecessors sets $P_{u'vd+1}$.

- Remove vertices with no incoming edges in L_{vH}.

For each vertex of G_v^H, either one arc is removed, either each outgoing arc is removed (at worse n). The number of predecessors of each vertex can be tracked during construction of G_v^H. As for each vertex, operations are performed at worse in $O(n)$, SPR is performed in the worst case in $O(n^2H)$.

Root neighbour reductions

In any path P rooted in v in G, the second vertex is in any case a neighbour of v. These neighbours of v can only appear in layer 1 in the representation of P on G_v^H when considering P is a shortest path. *Root Neighbour Reductions* (RNR) remove from G_v^H all vertices u_d such as $vu \in E$ and $d > 1$, as well as all vertices that have no more incoming arcs after removal of neighbours of v.

Lemma 6. *All vertices $u_d \in V_v^H$ such as $vu \in E$ and $d > 1$ are redundant.*

Proof. Consider a subgraph $G' = G(V')$, where $V' \subseteq V$ contains v, a shortest path P in G' rooted in v of maximum length H, its representation P^H on G_v^H and a vertex $u_d \in V_v^H$ with $d > 1$ such as $vu \in E$. Vertex $u_d \notin P^H$, otherwise P is not a shortest path as vertices from position 1 to $d-1$ could be eliminated. Thus vertex u_d is redundant. □

In Figure 4, considering the layered graph G_v^H of Figure 1, vertices 1 and 2 are eliminated from all layers except layer 1.

The RNR can be applied to a layered graph G_v^H as follows:

- For $d := 2$ to H:
  For each vertex $u \in L_{vd}$:
    If $vu \in E$ or $|P_{uvd}| = 0$:
      Remove u_d form V_v^H as well as all incoming and outgoing arcs $u_d u'_{d+1}$,
      Remove u from all predecessors sets $P_{u'vd+1}$.

As each node of G_v^H is inspected and at most $O(n)$ arcs are removed from all of them, RNR is performed in the worst case in $O(n^2H)$.

Triangle reductions

If an induced subgraph $G' \subseteq G$ contains some triangle $u\,u'\,u''$, a shortest path P from $v \in V$ to a another vertex of G' will not contain the three vertices of the triangle. If the first vertex of $u\,u'\,u''$ in P is at distance $d-1$, then the two other vertices are reachable at distance d. *Triangle Reductions* (TR) remove redundant arcs in triangles, as well as all vertices that have no more incoming arcs.

Lemma 7. *If G contains a triangle $u\,u'\,u''$, $\delta^-(u_d) = 1$ and $u''_{d-1}u_d \in A^H_v$, then arc $u_d u'_{d+1}$ is redundant.*

Proof. Assume there exists a triangle $u\,u'\,u''$ in G such that $\delta^-(u_d) = 1$, $u''_{d-1}u_d \in A^H_v$ and $u_d u'_{d+1} \in A^H_v$ is not redundant. Then there exists a subgraph $G' = G(V')$, where $V' \subseteq V$ contains v, and a shortest path P from v to u' in G' rooted in v such that arc $u_d u'_{d+1}$ belongs to P. Since $u''_{d-1}u_d$ is the only arc incoming in u_d, it also belongs to P. As $u\,u'\,u''$ form a triangle in G, $u''_{d-1}u'_d \in A^H_v$ and the path obtained by adding $u''_{d-1}u'_d$ to the subpath of P from v to u''_{d-1} is shorter than P, contradicting the fact that P is a shortest path. Hence $u_d u'_{d+1}$ is redundant. □

The TR can be applied as follows:

- For $d := 1$ to $H-1$:
  For each vertex u in L_{vd}:
    Check the number $|P_{uvd}|$ of incoming arc of u_d,
    If $|P_{uvd}| = 1$ and $d < H$:
      Let u'_{d-1} be the predecessor of u_d,
      For each arc $u_d u''_{d+1} \in A^H_v$ with $u' \neq u''$:
        If $uu'u''$ is a triangle in G:
          Remove $u_d u''_{d+1}$,
          Remove u from set $P_{u''vd+1}$.
    If $|P_{uvd}| = 0$:
      Remove u_d and all its outgoing edges $u_d u'_{d+1}$,
      Remove u from all predecessors sets $P_{u'vd+1}$.
- Remove vertices with no incoming edges in L_{vH}.

As each node of G^H_v must be inspected and at most $O(n)$ operations are performed for each of them to eliminate outgoing arcs, TR are performed in the worst case $O(n^2 H)$.

Figure 5 illustrates TR on the graph of Figure 1. In the initial graph, consider triangle 2-4-5. In the layered graph of G if 2_1 is in T^H_v, vertices 4_2 and 5_2 can be in T^H_v. There is no reason to consider path $2_1 - 5_2 - 4_3$ in G^H_v and as 5_2 can only be reached through 2_1, arc $(5_2, 4_3)$ will not appear in a shortest path from v.

Shortest path tree reductions

A combination of reductions SPR, RNR and TR, can be applied through *Shortest Path Tree Reductions* (SPTR) to reduce the size of G^H_v, leading to a graph G^{HR}_v. Set $G^{HR} = \bigcup_{j \in V_f} G^{HR}_j$. In reductions SPR and TR, a necessary condition to remove an arc is that its source must have only one incoming arc. After applying SPR or TR, the number of incoming arcs of some vertices decrease, leading to possible new arc elimination if reductions are performed once again. Consider as an example the layered graph from Figure 1. Arc $6_4 7_5$ is not eliminated when applying once

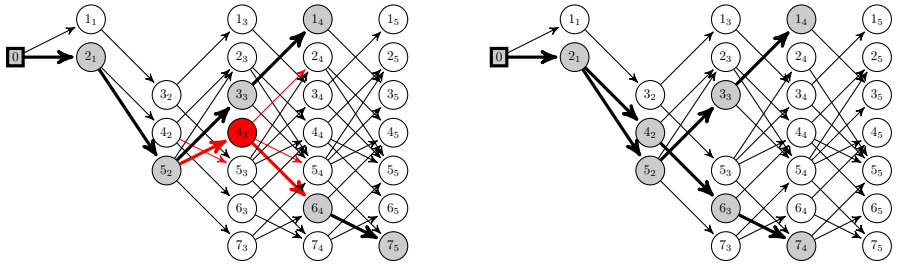

Figure 5: Triangle reductions

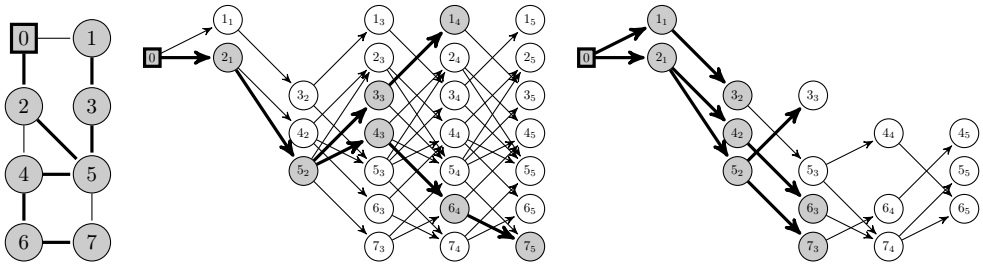

Figure 6: Shortest path tree reductions applied on a layered graph

SPR or TR in Figures 3 and 5. After applying SPR and TR, vertex 6_4 has only one incoming arc left, coming from 7_3, thus arc 6_47_5 will be removed if SPR are performed once again.

To perform SPTR, first RNR are applied before iterating SPR and TR until no more arc is removed. Performing an iteration of SPR and TR is made in $O(n^2H)$. If at an iteration of SPR-TR some arcs have been removed, consider the minimum distance d for which an arc $u_du'_{d+1}$ has been removed. Iterating SPR-TR is interesting in the case there are new vertices with only one incoming arc. As there are no new vertices with only one incoming arc in L_{vd} after the last iteration of SPR-TR, no arc between layers d and $d+1$ will be removed after a new iteration. This leads to a maximum of $H-1$ iterations of SPR-TR and a worst case complexity of $O(n^2H^2)$ for SPTR. When working with bipartite graphs, SPTR have a complexity of $O(n^2H)$ as, after RNR, only a single iteration of SPR is performed because there are no triangles to apply TR.

The complexity computed above is for applying reductions in a single layered graph. The overall complexity to apply reductions to all the layered graphs is thereofore $O(mn^2H^2)$.

Figure 6 gives the graph G of Figure 1 with layered graphs G_0^5 and G_0^{5R}. A total of 11 vertices (42%) and 29 arcs (63%) are eliminated with SPTR from the initial representation of G_0^H.

Lemma 8. *All feasible solutions of MMP can be represented on G^{HR}.*

Proof. Consider a feasible solution S of MMP and its components C_j^S. For each component C_j^S consider a shortest path tree T_j^S. As vertices and arcs removed from G_j^H with SPTR are redundant, T_j^S can be represented on G_j^{HR}. □

The solution spaces of LF and LFR are possibly reduced by SPTR as for each vertex i eliminated from L_{jd}, variable x_{ijd} is set to 0 with constraint (14) or (22). For each eliminated arc $i'_{d-1}i_d$,

the set of predecessors P_{ijd} of the target of this arc is reduced and its associated connectivity constraints (13) or (21) become tighter in both formulations as the right-hand side will contain less positive variables. In the following, we shall denote by LF-R, respectively LFR-R, formulation LF, respectively LFR, using G^{HR} rather than G^H. As for LF and LFR, LP-relaxations of LF-R and LFR-R are identical. Lemma 8 indicates that any feasible solution of LF-R, respectively LFR-R, is a feasible solution of LF, respectively LFR, thus $\mathcal{S}^{\text{LF-R}} \subseteq \mathcal{S}^{\text{LF}}$ and $\mathcal{S}^{\text{LFR-R}} \subseteq \mathcal{S}^{\text{LFR}}$, the same inclusion holds for LP-relaxations. A study of the computation times needed to solve different formulations is presented in Section 2.4.3. The objective values obtained when solving the LP-relaxations of LFR and LFR-R are compared in Section 2.4.3.

The reductions presented in this section can be applied to other hop constrained optimization problems, in particular if an optimal solution can be represented by a set of shortest path trees. For instance, SPR can be applied when searching a solution containing simple paths. On hop constrained spanning or Steiner trees [11, 13], the RNR and TR can also be used on instances where weight of arcs respect the triangular inequality, that is, the weight of an arc uv is at most the sum of weights of arcs of any path going from u to v. In the following section, we analyse the computation time needed to apply the reductions to layered graphs for MMP as well as the performance gain obtained by solving LF-R or LFR-R rather than LF or LFR.

2.4. Numerical results

After describing the instances used in this Section, we analyze the computation time needed to build layered graphs and perform reductions. The impact of the reductions on the size of the layered graphs is studied. We report the performances of VF, LF, LFR, LF-R and LFR-R, and their LP-relaxations, before analyzing the impact of parameters H, n and m.

All the solution methods were implemented using Java 1.8.0 and ILOG CPLEX 12.6 Java API. Tests were made on a 12-core i7-4930K 3.40 GHz processor limiting RAM memory to 4Gb and computation time to 1800s. The RAM is limited as none of the tests performed encounter memory issues. The average computation times provided include the time of instances that are not solved to optimality in 1800s.

2.4.1. Instances

Instances from [20] that were kindly provided by A. Rossi are used to compare VF, LF and LFR. Rossi et al. uses mainly two types of instances: *mimetic* and *square*. These instances are derived from grid-graphs [7]. A grid graph $G_{m,m}$ is a graph having vertices at all possible integer coordinates (x, y), with $0 \leq x < m$, $0 \leq y < n$. An edge uv, $u = (x_u, y_u)$ and $v = (x_v, y_v)$, is in $G_{m,m}$ if $u, v \in G_{m,m}$ and $|x_u - x_v| + |y_u - y_v| = 1$. Grid-graphs are bipartite graphs [7] and have a structure close to real electricity distribution networks [6]. We refer to mimetic instances as *bipartite* instances in the remainder of the paper. Additional bigger bipartite instances have been generated as well as *diagonal* instances to evaluate solving of MMP on non bipartite graphs. All instances are of the following types:

- *square*: G is a complete square grid-graph,
- *bipartite*: G is a connected subgraph of a *square* instance,
- *diagonal*: modification of *bipartite* instances, where some edges have been transformed into diagonals.

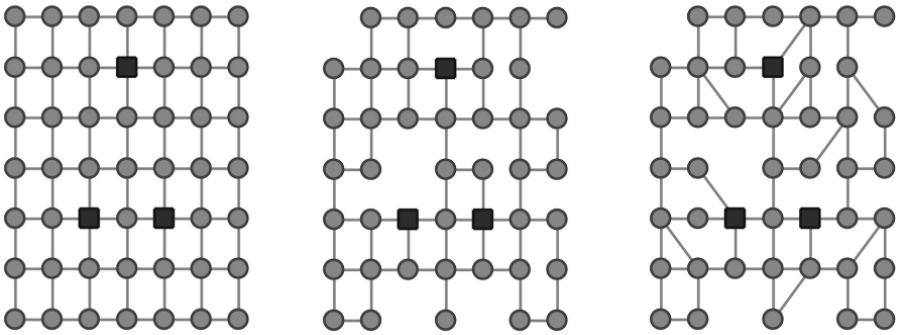

Figure 7: Overview of *square*, *bipartite* and *diagonal* instances

Diagonal instances can contain triangles or cycles of odd length, none of the diagonal instances used are bipartite.

Instances are illustrated on Figure 7, square nodes represent feeders. The feeders are placed on a circle centered in the middle of the graph. The density of bipartite and diagonal instances, representing the proportion of nodes that are kept from the original grid-graph, is on average 60%. All these instances are connected and the average degree of the nodes is between 2.6 and 2.9. The circle nodes are the terminal nodes and have an integer random demand between 0 and 100, terminal nodes with demand equal to 0 are the Steiner nodes of V_s. The capacity of the feeders is sufficient so that any feeder could supply the whole network to guaranty M_{min} is positive. Each configuration *type-n-m* is tested over 10 instances. For bipartite instances, those with up to 100 customers and 5 feeders are from Rossi et al. [20].

Lemma 9 allows us to consider only instances having the same capacity for all feeders.

Lemma 9. *An instance of MMP can be transformed in linear time into an instance where all feeders have the same capacity and M_{min} is identical.*

Proof. Consider an instance $\mathcal{I}$ of MMP and its graph $G = (V, E)$. An instance $\mathcal{I}'$ having the same capacity c for all feeders and its graph $G' = (V', E')$ can be built from $\mathcal{I}$ as follows:

1. Start with $G' = G$ and consider the feeder of $\mathcal{I}$ having the maximum capacity $c = \max\{c_j | j \in V_f\}$.
2. For each feeder $j \in V'_f$ such as $c_j < c$:
   (a) add a customer i to V' and an edge ij to E',
   (b) set demand of i to $c - c_j$,
   (c) set capacity of j to c.

From a feasible solution S' of $\mathcal{I}'$, a feasible solution S of $\mathcal{I}$ can be built be deleting the customers added during construction of $\mathcal{I}'$ from S'. As the customers added can be assigned to one feeder only, margins are identical in S and S' for each feeder. □

2.4.2. Shortest path tree reductions

Reductions SPR, RNR, TR and SPTR are tested on G^H of bipartite and diagonal instances with various values for parameters m, n and H. The proportion of vertices and arcs removed in the

Instance				G^H			SPR		RNR		TR		SPTR = G^{HR}			
type	n	m	H	time	vert.	arcs	vert.	arcs	vert.	arcs	vert.	arcs	time	it.	vert.	arcs
bip	100	10	6	0.01	622	1166	-9%	-28%	-13%	-24%	-	-	<0.01	1.0	- 23 %	- 45 %
	100	10	10	0.01	1705	3611	-7%	-25%	-10%	-16%	-	-	<0.01	1.0	- 20 %	- 40 %
	100	20	6	<0.01	913	1589	-15%	-35%	-22%	-32%	-	-	<0.01	1.0	- 35 %	- 54 %
	100	20	10	0.01	2335	4624	-17%	-35%	-20%	-26%	-	-	<0.01	1.0	- 39 %	- 56 %
	500	10	12	0.01	3671	8716	-2%	-14%	-6%	-10%	-	-	0.01	1.0	- 9 %	- 25 %
	500	10	16	0.02	6608	16254	-1%	-12%	-4%	-8%	-	-	0.01	1.0	- 7 %	- 21 %
	500	20	12	0.02	6406	14543	-3%	-17%	-6%	-10%	-	-	0.01	1.0	- 11 %	- 27 %
	500	20	16	0.04	12477	29617	-2%	-15%	-4%	-7%	-	-	0.03	1.0	- 7 %	- 22 %
diag	100	10	6	0.01	1258	2691	-7%	-23%	-21%	-31%	-6%	-13%	0.01	4.6	- 42 %	- 64 %
	100	10	10	0.02	3643	9039	-4%	-18%	-16%	-21%	-5%	-9%	0.02	5.7	- 36 %	- 56 %
	100	20	6	0.01	1782	3472	-13%	-31%	-34%	-43%	-8%	-14%	<0.01	3.9	- 58 %	- 74 %
	100	20	10	0.02	5160	11691	-10%	-27%	-29%	-34%	-7%	-12%	0.02	5.5	- 56 %	- 73 %
	500	10	12	0.03	8318	23141	-1%	-10%	-7%	-11%	-2%	-5%	0.07	6.4	- 13 %	- 31 %
	500	10	16	0.06	15810	45796	-1%	0%	1%	2%	3%	4%	0.15	6.4	- 9 %	- 24 %
	500	20	12	0.04	13407	33791	-2%	-13%	-8%	-11%	-3%	-6%	0.10	6.9	- 20 %	- 39 %
	500	20	16	0.09	26846	70878	-1%	-11%	-5%	-8%	-2%	-4%	0.25	7.3	- 13 %	- 29 %

Table 1: Elimination of vertices and arcs in layered graphs with SPR, RNR and TR

resulting layered graphs are given in Table 1. The computation time to generate layered graphs and perform reductions, as well as the number of iterations of SPTR, are also reported. The number vertices in the layered graphs G^H corresponds to the number of variables x_{ijd} in LF and LFR not initially fixed to zero. The computation time needed to generate G^H is similar to the time needed to perform the preprocessing needed to build VF. To perform SPTR in bipartite instances, as only one iteration of RNR and SPR is performed and complexities to build G^H and perform SPTR with a single iteration are identical, the time required is more or less the same as to build G^H. On diagonal instances, several iterations are made during SPTR, leading to a higher computation time and more vertices and arcs eliminated. As there are at most $H-1$ iterations during SPTR, the computation times stays reasonable. We observe in Section 2.4.3 that the preprocessing times for all formulations are negligible compared to computation times of the MIP.

On average, 25% of the vertices and arcs are removed by SPTR. The number of vertices and arcs removed with SPTR is generally bigger than applying SPR, RNR and TR separately, even for bipartite instances where a single iteration of SPTR is performed. This can be explained by the fact that some reductions enable some further reductions of other types. Also note that bipartite instances have less vertices and arcs than diagonal ones as expected with Lemma 1.

2.4.3. Comparison of the formulations

Table 2 compares the number of instances solved and the average computation times of all instances, including layered graph construction and preprocessing techniques of formulations VF, LF, LF-R, LFR and LFR-R. Values H used for hop constraints are the same as in Rossi et al. [20] and are all greater than or equal to d_{min}. The best computation time for each set of instances is indicated in bold. When VF is not solved to optimality, no feasible solution is found as it is a cutting plane algorithm. When not solved to optimality, all layered formulations found at least one feasible solution. The final gap of instances not solved to optimality is given in Table 3 with the LP-relaxation gap of different formulations. The formulation that has the best computation times is LFR-R for 11 of the 20 tested configurations. Figure 8 shows the proportion of solved instances with respect to time for each formulation. It is interesting to observe that although LFR(-R) has significantly less binary variables than LF(-R), it is beaten on almost half of the configurations by LF(-R). In Figure 8, we observe that LF is globally better than LFR. This can be explained

Instance				VF		LF				LFR			
						G^H		G^{HR}		G^H		G^{HR}	
Type	n	m	H	#sol.	time	#sol.	time	#sol.	time	#sol.	time	#sol.	time
bip	50	3	6	10	0.13	10	0.06	10	**0.04**	10	0.08	10	0.09
	50	5	6	10	1.46	10	0.13	10	0.16	10	0.26	10	**0.12**
	100	3	6	10	0.13	10	0.10	10	**0.09**	10	0.11	10	0.12
	100	5	6	10	0.84	10	0.13	10	0.14	10	0.15	10	**0.12**
squ	49	3	6	10	3.5	10	0.29	10	0.24	10	0.19	10	**0.18**
	49	5	6	10	130.05	10	12.86	10	10.68	10	10.00	10	**5.93**
	100	3	6	10	0.63	10	0.23	10	0.22	10	0.20	10	**0.15**
	100	5	6	10	52.17	10	1.71	10	**1.50**	10	2.32	10	1.75
bip	100	10	6	7	548.95	10	4.04	10	3.96	10	24.06	10	**3.86**
	100	20	6	10	143.85	10	0.37	10	**0.34**	10	0.63	10	0.39
	200	10	8	8	369.16	10	2.09	10	1.89	10	2.73	10	**0.89**
	200	20	8	10	195.59	10	1.07	10	0.85	10	2.04	10	**0.71**
	500	10	12	1	1756.87	8	394.43	8	**355.71**	8	441.25	9	398.48
	500	20	12	0	1800.0	7	583.48	7	**548.28**	7	561.42	7	551.91
diag	100	10	6	5	980.78	10	20.67	10	26.33	10	110.06	10	**18.14**
	100	20	6	8	437.07	10	0.52	10	**0.47**	10	1.26	10	1.01
	200	10	8	4	1087.69	9	291.4	9	233.69	9	296.16	10	**89.46**
	200	20	8	6	777.66	9	204.47	9	**183.92**	8	368.16	9	187.12
	500	10	12	0	1800.0	5	1070.78	5	1011.41	3	1402.0	5	**934.81**
	500	20	12	0	1800.0	4	1083.72	5	**1008.04**	4	1266.77	4	1089.53

Table 2: Comparaison of VF, LF and LFR

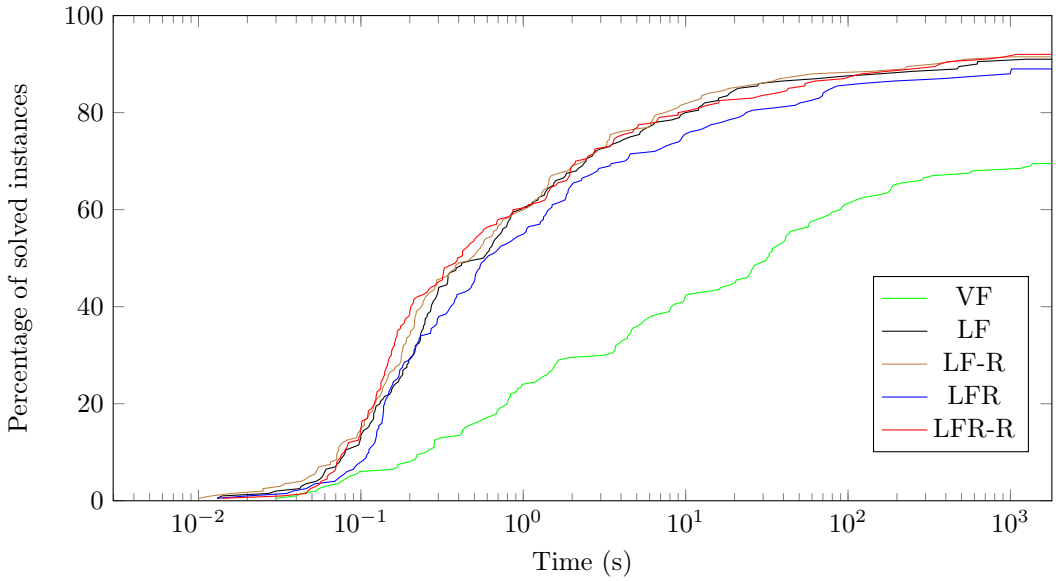

Figure 8: Percentage of solved instances with respect to time

by the powerful preprocessing and cutting techniques that have been developed for models using integer and binary variables, from Gomory's cuts end of the 50's [10] to recent work [8, 15], that are integrated in state-of-the-art MIP solvers. On average, the preprocessing of CPLEX eliminates 32% more variables in LF compared to LFR. Concerning the efficiency of SPTR, computation times are on average 23% lower for LF-R compared to LF and 47% lower for LFR-R compared to LFR. Formulations LF and LFR have globally much smaller computation time than VF, especially for larger instances where the number of connectivity and distance cuts that need to be added to VF gets much larger. A high standard deviation can be observed in Table 2 for layered formulations on computation times of some instances. When solving *diag-500-10-12* using LFR-R, five instances are not solved in 1800s and the other five instances are solved on average in 69s. It can also be observed that the computation times tend to be drastically smaller for bipartite instances than for diagonal instances. This can be a consequence of the small number of vertices in layered graphs of bipartite graphs as shown in Lemma 1.

Table 3 reports the gaps of the LP-relaxations of each formulation as well as the gap after solving the root node of the branch & bound tree (with cuts added by CPLEX) and the final gaps of instances not solved to optimality.

Instance				Best obj.	LF				LF-R				Closed gap		
Type	n	m	H		LP gap	root node gap	#sol.	final gap	LP gap	root node gap	#sol.	final gap	LP gap	root node gap	final gap
bip	100	10	6	1136.8	12.62	1.82	10	-	12.39	1.71	10	-	2 %	4%	-
	100	20	6	414.8	100.17	2.07	10	-	76.82	1.7	10	-	23%	17%	-
	200	10	8	2376.6	34.94	2.85	10	-	24	2.14	10	-	31%	23%	-
	200	20	8	1000.4	71.13	5.4	10	-	26.11	8	10	-	63%	-47%	-
	500	10	12	6741.2	21.57	6.07	8	1.35	13.48	4.72	8	1.28	38%	17%	5%
	500	20	12	2938	65.81	8.23	7	2.94	40.84	7.93	7	2.64	37%	1%	15%
diag	100	10	6	1084.9	37.54	1.95	10	-	32.44	1.91	10	-	13%	2%	-
	100	20	6	442	72.58	4.64	10	-	56.75	4.64	10	-	21%	0%	-
	200	10	8	2383.5	27.65	4.53	9	2.38	11.44	4.49	9	1.67	58%	-1%	29%
	200	20	8	1029.4	42.21	13.38	9	1.75	36.03	10.91	9	1.22	14%	18%	50%
	500	10	12	6450.7	10.32	4.52	5	2.58	7.74	3.87	5	1.29	26%	17%	40%
	500	20	12	3038.1	67.31	18.53	4	6.38	62.89	17.92	5	4.25	7%	3%	32%

Instance				Best obj.	LFR				LFR-R				Closed gap		
Type	n	m	H		LP gap	root node gap	#sol.	final gap	LP gap	root node gap	#sol.	final gap	LP gap	root node gap	final gap
bip	100	10	6	1136.8	12.85	4.55	10	-	12.39	4.09	10	-	4 %	10 %	-
	100	20	6	414.8	104.03	11.37	10	-	76.82	5.48	10	-	26 %	52 %	-
	200	10	8	2376.6	35.89	5.47	10	-	24	5.23	10	-	33 %	4 %	-
	200	20	8	1000.4	70.73	16.01	10	-	26.11	9.7	10	-	63 %	39 %	-
	500	10	12	6742.3	22.92	6.07	8	1.35	13.48	6.74	9	1.35	41 %	-11 %	0 %
	500	20	12	2938.4	73.17	12.05	7	10.87	41.43	10.58	7	6.17	43 %	12 %	43 %
diag	100	10	6	1084.9	37.54	12.26	10	-	32.76	10.96	10	-	13 %	11 %	-
	100	20	6	442	73.77	31.91	10	-	56.75	6.14	10	-	23 %	81 %	-
	200	10	8	2385.1	26.95	7.63	9	5.25	11.45	6.2	10	-	58%	19 %	100 %
	200	20	8	1048.8	43.94	15.42	8	9.23	36.71	10.38	9	5.03	16 %	33 %	46 %
	500	10	12	6450.1	10.32	6.45	3	3.87	8.39	4.52	5	2.58	19 %	30 %	33 %
	500	20	12	3028.3	84.09	21.01	4	306.46	72.07	18.17	4	23.92	14 %	14 %	92 %

Table 3: Gap analysis of layered formulations

The «Best obj.» column reports the average objective of the best solutions found over all formulations. Final gap averages are computed over unsolved instances only. As the capacity of the feeders in the tested instances is arbitrarily high, the objective value of MMP is also arbitrarily high. Therefore we report absolute gaps rather than relative gaps that are all very low. The absolute gap is the difference between the upper bound and the objective value of the best feasible solution found over all formulations. In order to interpret these values, it can be related to the number n of customers and the average demand of 50 for each customer. The «Closed gaps» columns report the proportion of the gap of G^H that is closed with G^{HR}. On average, 29% of the LP gap is closed, going up to 63% closed at best, while 45% of the final gap is closed for unsolved instances.

2.4.4. Impact of the hop limit value and of the size of the network

The value of the objective function and the computation time depending on the value of H is now analysed using LFR-R. Table 4 and 5 gives respectively for small and big instances the number of instances solved, the average evolution of the objective value, the computation times and the average final gaps for $H \in \{d_{min}, ..., d_{min} + 5\}$.

	Instance				$H = d_{min}$			+1			+2			+3			+4			+5		
	type	n	m	d_{min}	#sol	obj.	time	#sol	obj.	time	#sol	obj.	time	#sol	obj.	time	#sol	obj.	time	#sol	obj.	time
VF	bip	50	3	5.8	10	1676.3	0.12	10	+12.5	0.64	10	+0.9	1.78	10	+0.2	9.06	10	+0	13.92	10	+0	81.08
		50	5	5.2	10	1999.3	0.35	10	+41.8	1.34	10	+2.6	8.96	10	+4.2	38.23	10	+0	203.71	8	-	578.03
		100	3	6	10	3403.9	0.11	10	+0	1.37	10	+0	5.86	10	+0	57.6	10	+0	212.63	7	-	799.28
		100	5	5.9	10	3998	0.73	10	+59.6	31.33	10	+33.1	210.5	9	-	393.51	7	-	981.39	3	-	1367.42
	squ	49	3	4	10	1625.2	0.05	10	+0.6	0.56	10	+0	3.47	10	+0	8.35	10	+0	63.54	10	+0	401.2
		49	5	3	10	1944.4	0.12	10	+5.8	4.91	10	+0.7	34.43	10	+0	130.24	9	-	549.21	3	-	1503.36
		100	3	6	10	3336.3	0.66	9	-	186.3	10	+0	61.3	10	+0	586.1	4	-	1617.35	2	-	1630.87
		100	5	5	10	4003.4	100.57	10	+0	52.27	10	+0	552.53	0	-	1800	0	-	1800	1	-	1794.92
	Averages				10	2748.35	12.83	9.8	-	34.84	10	+4.66	109.85	8.6	-	377.88	7.5	-	680.21	5.5	-	1019.52
LFR-R	bip	50	3	5.8	10	1676.3	0.06	10	+12.5	0.12	10	+0.9	0.16	10	+0.2	0.2	10	+0	0.22	10	+0	0.41
		50	5	5.2	10	1999.3	0.07	10	+41.8	0.1	10	+2.6	0.18	10	+4.2	0.39	10	+0	0.65	10	+0	1.06
		100	3	6	10	3403.9	0.13	10	+0	0.14	10	+0	0.19	10	+0	0.26	10	+0	0.43	10	+0	0.7
		100	5	5.9	10	3998	0.15	10	+59.6	0.45	10	+33.1	1.02	10	+1.1	4.89	10	+1.1	17.38	10	+0.1	29.79
	squ	49	3	4	10	1625.2	0.1	10	+0.6	0.11	10	+0	0.19	10	+0	0.31	10	+0	0.53	10	+0	1.01
		49	5	3	10	1944.4	0.08	10	+5.8	0.41	10	+0.7	3.88	10	+0	5.94	10	+0	21.32	10	+0	26.93
		100	3	6	10	3336.3	0.16	10	+0	0.31	10	+0	0.56	10	+0	1.11	10	+0	2.66	10	+0	5.79
		100	5	5	10	4003.4	0.64	10	+0	1.73	10	+0	6.54	10	+0	4.93	10	+0	33.16	10	+0	47.9
	Averages				10	2748.35	0.17	10	+15.03	0.42	10	+4.66	1.59	10	+0.68	2.25	10	+0.13	9.54	10	+0.01	14.19

Table 4: H impact solving VF and LFR-R on small instances

	Instance				$H = d_{min}$				+1				+2			
	type	n	m	d_{min}	#sol	obj.	final gap	time	#sol	obj.	final gap	time	#sol	obj.	final gap	time
LFR-R	bip	100	10	4.8	10	1065.3	-	0.34	10	+65.4	-	2.67	10	+21.2	-	12.99
		100	20	4.4	10	329.8	-	0.08	10	+62.9	-	0.17	10	+48.8	-	0.41
		200	10	7.1	10	2326.1	-	0.19	10	+57.8	-	0.63	10	+83.7	-	12.7
		200	20	5.5	10	879.3	-	0.12	10	+46.3	-	0.22	10	+60.5	-	0.42
		500	10	9.8	10	6563.2	-	3.46	10	+59.2	-	103.17	8	+108	1.31	474.63
		500	20	8.3	10	2543.8	-	11.7	10	+170.3	-	33.1	8	+95.5	7.88	364.4
	diag	100	10	4.4	10	995.8	-	0.07	10	+57.4	-	2.43	10	+47.4	-	28.22
		100	20	3.9	10	365.7	-	0.09	10	+59.9	-	0.28	10	+16.4	-	0.86
		200	10	5.9	10	2205.7	-	0.17	10	+108.7	-	3.53	10	+71.9	-	88.58
		200	20	5.6	10	843.4	-	0.16	10	+79.2	-	0.73	10	+98.4	-	6.27
		500	10	8.5	10	6224.9	-	8.3	9	+109.6	1.87	248.03	6	+50.0	1.24	894.07
		500	20	8.5	9	2654.8	4.24	180.87	9	+169.2	3.45	188.75	7	+141.4	6.37	686.55
	Averages				9.9	2249.8	4.26	17.13	9.8	+87.2	2.66	48.64	9.1	+70.3	4.02	214.18

	Instance				+3				+4				+5			
	type	n	m	d_{min}	#sol	obj.	final gap	time	#sol	obj.	final gap	time	#sol	obj.	final gap	time
LFR-R	bip	100	10	4.8	10	+0.1	-	68.19	10	+28.9	-	147.63	8	+0.1	2.23	415.2
		100	20	4.4	10	+11.2	-	1.71	10	+0	-	4.64	10	+0	-	15.1
		200	10	7.1	9	+11.6	1.16	242.56	10	+17.1	-	84	10	+0.7	-	184.95
		200	20	5.5	10	+62	-	1.1	10	+8.2	-	2.86	10	+5	-	11.31
		500	10	9.8	8	+51.4	1.31	559.23	7	+30.6	1.99	599.81	6	+15	1.97	1162.46
		500	20	8.3	7	+109	3.05	551.28	6	+34.3	5.85	723.24	4	+39.2	9.16	1081.88
	diag	100	10	4.4	9	+1.9	1.39	266.77	7	-0.2	3.98	601.43	6	-0.1	2.88	728.96
		100	20	3.9	10	+0.7	-	3.6	10	+0.4	-	10.16	9	+0	1.53	193.19
		200	10	5.9	9	+33.7	4.85	439.91	8	+21.2	3.08	641.55	8	+10	5.73	548.54
		200	20	5.6	9	+24.3	4.04	188.72	9	+0	8.26	222.57	8	-0.4	9.69	399.41
		500	10	8.5	6	+62.6	1.86	848.77	3	+15.8	2.48	1503.62	0	+3.1	3.11	1800
		500	20	8.5	4	+68.4	9.02	1089.59	4	+0.7	21.50	1113.25	4	-8.7	73.80	1175.65
	Averages				8.4	+36.4	4.21	355.12	7.8	+13.1	8.27	471.23	6.9	+5.3	13.66	643.06

Table 5: H impact solving LFR-R on large instances

Figure 9 gives the proportion of solved instances depending on time for all values of H used. In this Figure, instances used are the bipartite and diagonale ones with at least 100 terminal nodes and 10 feeders. As in Table 3, final gaps in Table 5 are absolute gaps and computed only over

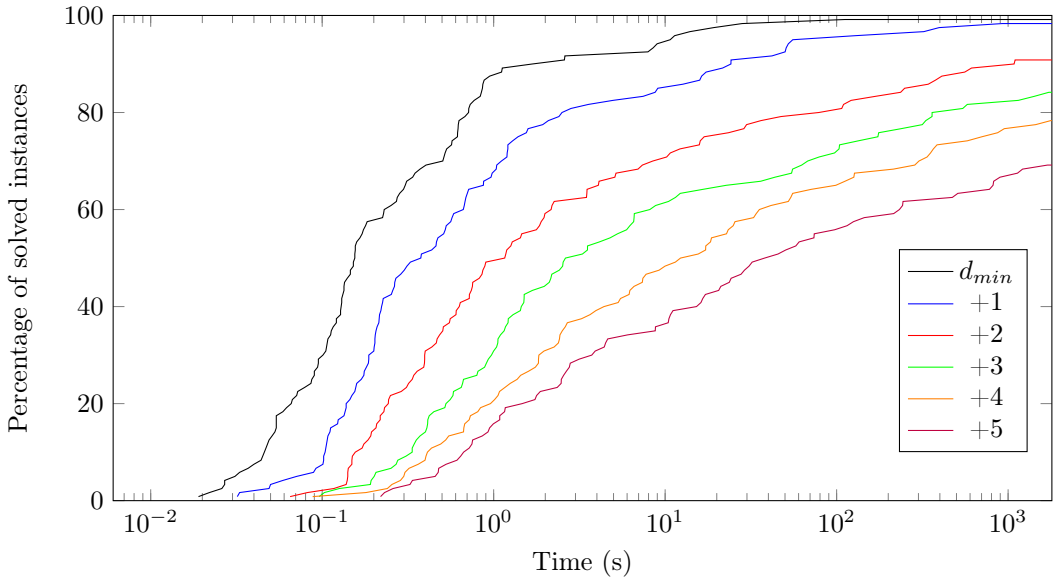

Figure 9: Percentage of instances solved depending on time with LFR on G^{HR}

the unsolved instances. As expected, the computation times strongly increase and the optimal value increases with H increasing. Formulation VF cannot solve all its instances from $d_{min} + 3$ while formulation LFR-R solves the same instances in a short time up to $d_{min} + 5$. Concerning tests performed on bigger instances, the variation of the objective value strongly depends on the type of instance. For some unsolved instances with LFR-R for a big value H, the best feasible solution tends to get worse than the best feasible solution found for $H - 1$, as for *diag-100-10* with $H = d_{min} + 4$ or $+5$. It can also be noticed that for some unsolved instances, the final gap stays very low. Consider instances *diag-500-10* with $H = d_{min} + 5$, no instance is solved although the absolute final gap is on average 3.1.

Figure 10 gives the computation times with different values of n and m with $H = d_{min} + 3$. For each curve, ten bipartite and ten diagonal instances are used. On the left figure, computation times increase as the number of clients increases. On the right figure, computation times do not necessarily increase as the number of feeders increases, computation times with 20 feeders are smaller than those with 10 feeders. This can be explained as d_{min} can get smaller as the number of feeders increases and thus reduces the value of H.

3. Distribution networks with hop losses

3.1. Problem definition

Distribution networks can incur losses during transportation, as it is the case in electricity networks. This is one of the motivation of Rossi et al. [20] for adding hop constraints to MMP.

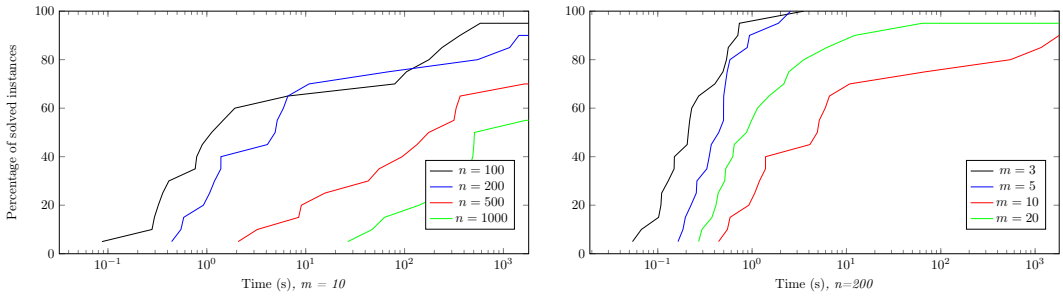

Figure 10: Impact of n an m solving LFR on G^{HR}

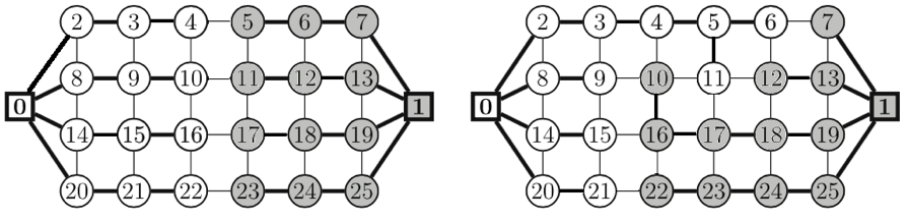

Figure 11: Truncated optimal solutions of MMP considering hop losses

Indeed, limiting the distance between feeders and customers indirectly limits the power loss.

Our objective in this section is to present a model allowing to explicitly consider losses due do transportation. Consider the MMP on the graph in Figure 11. All terminal nodes are customers with a demand equal to 1, $c_0 = c_1 = 24$ and $H = 5$, two optimal solutions of MMP are given with $M_{min} = 12$.

The second solution contains longer transportation distances and is worse than the first one when considering power losses. If there is a 5% loss of power per hop, the effective minimum margin 10.70 for the first solution and 10.38 for the second one. Solutions of MMP may have different effective minimum margins considering losses.

Let us assume we are given a *loss function* l that indicates the amount by which the demand of a customer assigned at distance d must be multiplied to compensate the power loss. It is a discrete increasing function such that $l(0) = 1$. If a customer is assigned at distance d of its feeder, its *adapted demand* is $l(d)dem_i$. The power losses are independent and only depend on the distances at which customers are assigned. We define the *Minimum Margin Problem considering Losses* (MMP_L) as the problem of finding a feasible solution of HDNCP maximizing the minimum margin of the feeders M^L_{min} that takes in consideration the power losses due to transportation. The margin of a feeder is now defined as the difference between its capacity and the sum of the adapted demands of customers assigned to it at specific distances. In the special case where $l(d) = 1$ for all $1 \leq d \leq H$, MMP_L reduces to MMP.

3.2. Reformulation of LFR

If MMP_L is solved with LFR, the optimal value obtained only provides an upper bound on the optimal minimum margin of MMP_L. In order to model the effective margins of the feeders taking into consideration the losses due to transportations, margin constraints (17) are replaced by

$$c_j - \sum_{i \in N_j} \sum_{1 \leq d \leq H} x_{ijd} dem_i l(d) \geq M^L_{min} \ \forall j \in V_f$$

leading to the *Layered Formulation Relaxed with Losses (LFRL):*

$$\text{Max } M^L_{min} \tag{25}$$

$$\text{s.t. } c_j - \sum_{i \in N_j} \sum_{1 \leq d \leq H} x_{ijd} dem_i l(d) \geq M^L_{min} \qquad \forall j \in V_f \tag{26}$$

$$\sum_{j \in V_f} x_{ij} = 1 \qquad \forall i \in V_c \tag{27}$$

$$\sum_{j \in V_f} x_{ij} \leq 1 \qquad \forall i \in V_s \tag{28}$$

$$x_{ij} = \sum_{1 \leq d \leq H} x_{ijd} \qquad \forall j \in V_f, i \in N_j \tag{29}$$

$$x_{ijd} \leq \sum_{k \in P_{ijd}} x_{kj(d-1)} \qquad \forall j \in V_f, d \in \{2, \ldots, D_{max}\}, i \in L_{jd} \tag{30}$$

$$x_{ijd} = 0 \qquad \forall j \in V_f, d \in \{1, \ldots, H\}, i \notin L_{jd} \tag{31}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in V_f, i \in N_j \tag{32}$$

$$0 \leq x_{ijd} \leq 1 \qquad \forall j \in V_f, d \in \{1, \ldots, D_{max}\}, i \in L_{jd} \tag{33}$$

Formulation LFRL applied G^{HR} is denoted LFRL-R. As power losses are considered, some symmetries in MMP are broken in MMP_L as illustrated in Figure 11, meaning potentially better computation times.

3.3. Numerical results

We tested LFRL on the same instances as in Section 2.4. We used the loss function $l(d) = (1-p)^{-d}$ that corresponds to a loss of a fraction p per hop. Table 6 presents the number of instances solved and the average computation times for algorithm LFRL-R with $H = d_{min} + 3$ for different values of p. Figure 12 shows the percentage of instances solved depending on time for all tested values of p.

The computation times generally decrease as p increases. This is probably explained by the fact that assigning a customer to different feeders results in different margins as the distance is likely not the same to each feeder. This breaks a lot of symmetries in the branching tree and therefore reduces the computing time.

Instance				$p = 0$		$p = 0.02$		$p = 0.05$	
type	n	m	H	#sol	time	#sol	time	#sol	time
bip	50	3	6	10	0.09	10	0.07	10	0.06
	50	5	6	10	0.11	10	0.15	10	0.09
	100	3	6	10	0.12	10	0.13	10	0.16
	100	5	6	10	0.13	10	0.22	10	0.15
squ	49	3	6	10	0.18	10	0.21	10	0.20
	49	5	6	10	5.91	10	4.83	10	1.20
	100	3	6	10	0.15	10	0.15	10	0.14
	100	5	6	10	1.74	10	1.52	10	1.26
bip	100	10	6	10	3.86	10	1.81	10	1.51
	100	20	6	10	0.39	10	0.35	10	0.29
	200	10	8	10	0.92	10	1.14	10	0.55
	200	20	8	10	0.7	10	0.62	10	0.45
	500	10	12	9	398.84	10	360.01	10	9.31
	500	20	12	7	551.91	7	577.06	7	544.96
diag	100	10	6	10	28.17	10	8.83	10	7.39
	100	20	6	10	1.01	10	0.88	10	1.13
	200	10	8	10	89.17	9	185.05	10	7.69
	200	20	8	9	187.08	10	123.55	10	33.71
	500	10	12	5	1033.42	6	889.53	6	895.27
	500	20	12	4	1089.46	4	1083.08	4	1081.99
Averages				9.2	169.668	9.3	161.9595	9.35	129.37

Table 6: p impact solving LFRL-R with $H = d_{min} + 3$

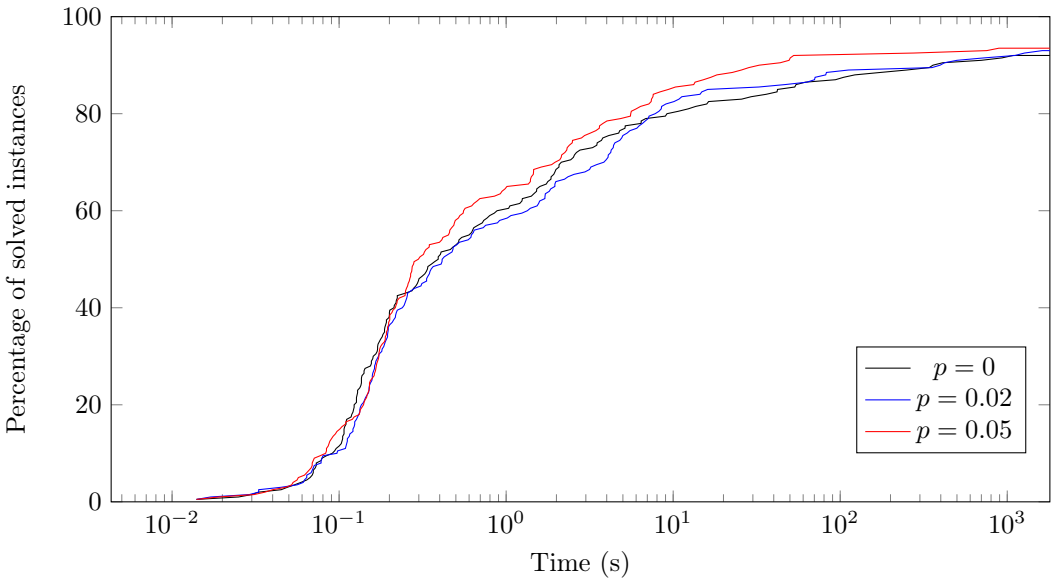

Figure 12: Percentage of instances solved depending on time

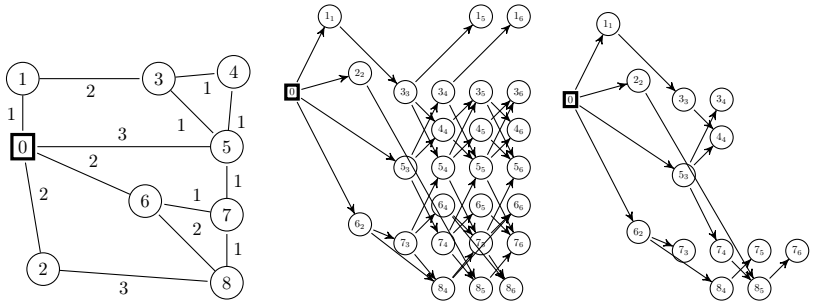

Figure 13: Layered graph of an edge-weigted graph G with $D = 6$ with and without SPTR

4. Distance constrained model

Another application of distribution network design that can be modeled as a variant of HDNCP arises in the context of multicast routing in telecommunications. In multicast routing, feeders send data to terminal nodes that have a given demand in data. These nodes are also used as relay to transfer data to other terminal nodes. Some delay is incurred by traversing the path from the feeder to the terminal node. In applications involving streaming data (such as audio or video), such delays must be limited [17].

Delays are often proportional to the physical length of links between nodes. We define the Multicast Routing Problem (MRP) as a variant of HDNCP where hop constraints are replaced by *delay constraints*. We consider a graph $G = (V, E)$ where edges have an integer length in $\{1, \ldots, L\}$ such that the triangular inequality is satisfied, and we want to impose a maximum length D on each path linking a feeder to a terminal node. Considering as before the minimum margin as objective to maximize, we denote this variant as MMP_D.

To solve MMP_D, we can use formulations introduced before by using a modified definition of layered graphs taking arc lengths into account. We construct a layered graph G_v^D rooted in v such that if an edge has length d, its endpoints will appear in layers that are separated by $d - 1$ layers rather than in consecutive layers, as illustrated on Figure 13.

When solving MMP_D, the layered graph reductions SPR can be applied, as well as RNR and TR if the triangular inequality is satisfied, leading to reduced layered graphs G_v^{DR}. Models LF(-R) and LFR(-R) can be used to solve MMP_D.

We evaluate the efficiency of LFR on reduced layered graphs $\{G_j^{DR}\}_{j \in V_f}$ (LFR-DR). The bipartite and diagonal instances of Section 2.4.1 are used with random lengths in $\{1, \ldots, L\}$ for each edge such that triangular inequality is satisfied. The limit on the path lengths is set to $D = d_{\min} + \lceil \frac{3L}{2} \rceil$, where $d_{\min}$ is the minimum feasible (weighted) distance. Table 7 reports the number of solved instances and the average computation times for different values of L and Figure 14 presents the percentage of solved instances depending on time.

The computation times increase strongly for values of L from 1 to 10 but tend to increase slower as the value of L increases, as observed on Figure 14.

Instance			$L=1$		$L=2$		$L=10$		$L=20$		$L=30$		$L=40$		$L=50$	
type	n	m	#sol	time	#sol	time	#sol	time	#sol	time	#sol	time	#sol	time	#sol	time
bip	100	10	10	2.98	10	154.24	9	248.11	8	454.33	8	373.77	9	350.01	7	679.23
	100	20	10	0.15	10	11.37	10	41.76	10	15.42	10	23.08	10	14.11	10	76.06
	200	10	10	0.66	10	22.87	9	232.05	8	521.28	8	572.36	7	439.65	9	270.08
	200	20	10	0.21	10	1.11	10	3.90	10	14.44	10	15.78	10	26.05	10	149.85
diag	100	10	10	2.79	10	83.76	8	494.86	6	551.41	6	727.19	7	805.03	7	627.82
	100	20	10	0.28	10	0.51	10	14.32	10	11.54	10	37.66	10	28.72	9	208.08
	200	10	10	3.71	9	263.09	7	831.05	6	823.87	5	949.25	6	942.92	5	1085.47
	200	20	10	0.78	9	192.73	8	369.27	8	442.07	9	311.26	8	377.60	8	367.64
Averages			10	0.98	9.8	91.21	8.9	279.41	8.3	354.28	8.3	376.29	8.4	373.01	8.1	433.03

Table 7: Time solving with LFR using various edge length L

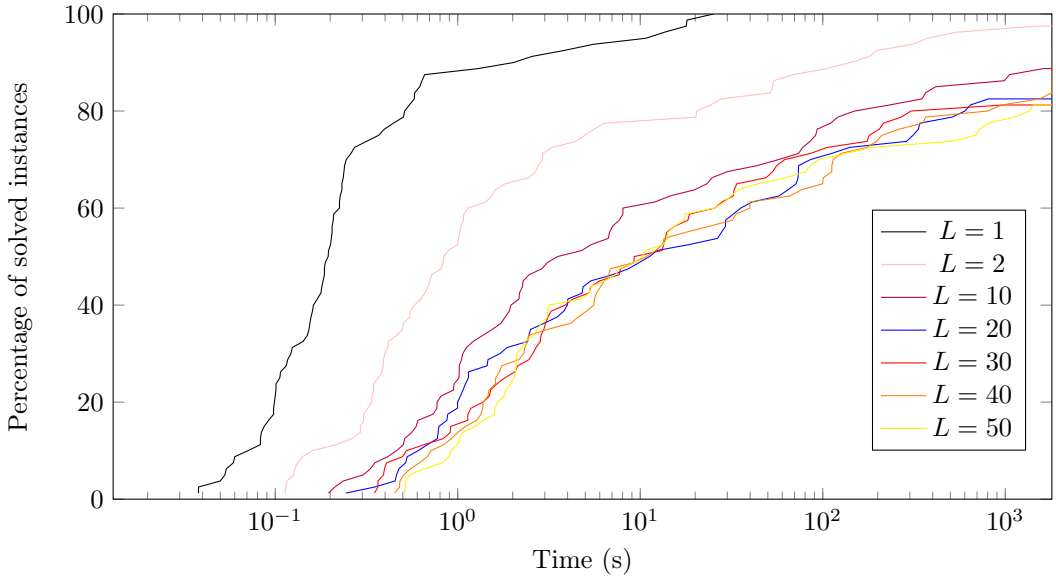

Figure 14: Percentage of instances solved depending on time

5. Conclusion

In this paper, we present efficient formulations and preprocessing techniques to solve the Hop Constrained Minimum Margin Problem and some of its variants. Our methods use layered graphs to derive stronger and more compact descriptions of the solution space using extended formulations. From our experiments, we can conclude that layered graphs, when used in conjunction with reduction techniques, lead to good extended formulations of hop constrained problems.

Another important lesson from our experiments is that relaxing a large number of binary variables of a MIP formulation does not necessarily improve the computation time using state-of-the-art MIP solvers. This is due to the efficiency of multiple preprocessing techniques and cutting-plane methods available in MIP solvers for dealing with binary variables.

Additional reductions could be studied and adapted to hop constrained problems with a different connectivity structure than those studied in this paper, e.g. edge-disjoint paths [2], or to deal with formulations where edge variables need to be created explicitly (e.g. hop constrained Steiner tree problems [12].

Acknowledgements

The authors thank the three anonymous referees for their useful comments on the paper. This work is supported by the Interuniversity Attraction Poles Programme P7/36 «COMEX» initiated by the Belgian Science Policy Office.

References

[1] A Balakrishnan and K Altinkemer. Using a hop-constrained model to generate alternative communication network designs. *ORSA J. Comput.*, 4(2):192–205, 1992.

[2] Quentin Botton, Bernard Fortz, Luis Gouveia, and Michael Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, 25:13–26, 2013.

[3] V. Chvátal. *Linear Programming*. Freeman and Company, New York, 1983.

[4] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8:1–48, 2010.

[5] Michel Crappe. *Stabilité et sauvegarde des réseaux électriques*. Hermès, 2003.

[6] B. Enacheanu, M.C. Alvarez., B. Raison., R. Caire, W. Bienia, O. Devaux, and N. Hadj-Said. Optimal meshed distribution network configuration. *International Review of Electrical Engineering*, 4(5):957–966, 2009.

[7] A. Itai et al. Hamilton paths in grid graphs. *Society for Industrial and Applied Mathematics*, 11(4):676–686, 1982.

[8] Gerald Gamrath, Thorsten Kock, Alexander Martin, Matthias Miltenberger, and Dieter Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015.

[9] M Garey and D Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completness*. W.H. Freeman and Company, New York, 1979.

[10] R. Gomory. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. *Bulletin Of the American Mathematical Society*, 64:275–278, 1958.

[11] Luis Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS Journal on Computing*, 10(2):180–188, 1998.

[12] Luis Gouveia, Markus Leitner, and Ivana Ljubić. Hop constrained steiner trees with multiple root nodes. *European Journal of Operational Research*, 236:100–112, 2014.

[13] Luis Gouveia, Luidi Simonetti, and Eduardo Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Mathematical Programming*, 128:123–148, 2011.

[14] Ivana Ljubić and Stefan Gollowitzer. Modelling the hop constrained connected facility location problem on layered graphs. *Electronic Notes in Discrete Mathematics*, 36:207–214, 2010.

[15] Hao-Chunn Lu, Yu-Chien Ko, and Huang Yao-Huei. A note on "reducing the number of binary variables in cutting stock problems". *Optimization Letters*, 8:569–579, 2014.

[16] Donella Meadows, Jorgen Randers, and Dennis Meadows. *Limits to Growth, the 30-Year Update*. Chelseq Green Publishing Company, 2004.

[17] Carlos Oliveira and Panos Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers and Operations Research*, 32:1953–1981, 2005.

[18] Hasan Pirkul and Samit Soni. New formulations and solution procedures for the hop constrained network design problem. *European Journal of Operational Research*, 148:126–140, 2003.

[19] Jeremy Rifkin. *The Third Industrial Revolution*. St. Martin's Griffin, 2011.

[20] André Rossi, Alexis Aubry, and Mireille Jacomino. Connectivity-and-hop-constrained design of electricity distribution networks. *European Journal of Operational Research*, 218:48–57, 2011.

[21] Stefan Voß. The steiner tree problem with hop constraints. *Annals of Operations Research*, 86:321–345, 1999.

[22] Craig Wigginton. Telecommunications industry outlook. http://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/telecommunications-industry-outlook.html, 2016.

[23] Laurence Wolsey. *Integer Programming*. Wiley-Interscience Publication, 1998.