Toward Scalable Algorithms for the Unsplittable Shortest Path Routing Problem

Amal Benhamiche, Morgan Chopin

Toward Scalable Algorithms for the Unsplittable Shortest Path Routing Problem

Amal Benhamiche
Data & Artificial Intelligence,
Orange Labs,
Châtillon, France,
amal.benhamiche@orange.com

Morgan Chopin
Data & Artificial Intelligence,
Orange Labs,
Châtillon, France,
morgan.chopin@orange.com

***Abstract*—In this paper, we consider the *Delay Constrained Unsplittable Shortest Path Routing* problem which arises in the field of *traffic engineering* for IP networks. This problem consists, given a directed graph and a set of commodities, to compute a set of routing paths and the associated administrative weights such that each commodity is routed along the unique shortest path between its origin and its destination, according to these weights. We present a compact MILP formulation for the problem, extending the work in [5] along with some valid inequalities to strengthen its linear relaxation. This formulation is used as the bulding block of an iterative approach that we develop to tackle large scale instances. We further propose a dynamic programming algorithm based on a *tree decomposition* of the graph. To the best of our knowledge, this is the first exact combinatorial algorithm for the problem. Finally, we assess the efficiency of our approaches through a set of experiments on state-of-the-art instances.**

***Index Terms*—Traffic engineering, IP networks, Mixed Integer Linear Programming, Dynamic Programming, Treewidth, Algorithms**

I. Introduction

In spite of the promises of the MPLS[1] forwarding scheme, most IP networks still heavily rely on shortest path rules where weights are assigned to links by network administrators and the routers are then able to compute shortest routing paths [14]. At the same time, the growing interest for content and user services driving even more traffic stresses the need to optimize the utilization of the available resources and maintain a high level of QoS on operational networks. On another hand, the arising of Network Virtualization will be a key enabler for the deployment of virtualized components capable of performing efficient path computation on behalf of the routers, thus allowing the optimization of operational IP networks. This perspective change draws again the traffic engineering community's attention to classical problems related to IP network optimization and raises the question of finding effective algorithms allowing to solve those problems for large scale networks.

The problem of finding routing weights inducing shortest paths that minimize the network congestion has been widely studied in the literature for both its *splittable* and *unsplittable* versions. Some early works addressing shortest path routing issues in an IP network optimization include [6]. The authors of this paper study the problem of designing a survivable VPN-based network using OSPF[2] routing protocol and propose a compact MILP formulation and several heuristics to solve the problem. In [9], the authors investigate the OSPF weights optimization problem with splittable traffic and a piecewise approximation of the load function. They show that the problem is NP-hard for a given set of demands and provide a local search heuristic to solve it. The problem is also shown NP-hard to approximate for both splittable and unsplittable versions (see [9] and [3]). The authors in [1] and [2] adress the unsplittable shortest path routing problem and study the properties of path sets that induce shortest path routing with compatible weights. Approaches based on Mixed Integer Linear programming for the problem include the work in [13], [5], [4]. In particular, compact formulations are proposed in [13] and [5] for the splittable version of the problem. Bley propose in [5] the so-called *two-phase algorithm*, an exact approach based on a decomposition of the problem into a master subproblem and a client subproblem. The former subproblem generates routing paths while the latter returns compatible weights if any, or *conflict inequalities* forbidding incompatible routing paths otherwise.

In this paper, we consider a variant of the *Unsplittable Shortest Path Routing* (USPR) problem with end-to-end delay constraints motivated by practical QoS requirements for the traffic. Our work extends the results proposed by [5] and [4]. Our main contributions are (i) a compact MILP formulation for the problem along with two classes of valid inequalities to strengthen its linear relaxation, (ii) a MILP-based heuristic that iteratively reroutes a portion of the traffic fixed by the decision-maker and reduces the network load and (iii) a dynamic programming algorithm based on a tree decomposition of the graph. To the best of our knowledge, this is the first exact combinatorial algorithm for the problem.

Our algorithms are designed with the two following objectives

[1] *MultiProtocol Label Switching*

[2] Open Shorest Path First

- **scalability**: they can be used to push back the limits of existing approaches for the problem in terms of size of instances treated,
- **flexibility**: they can be parameterized to provide solutions that modify only part of the traffic routing, which is highly desirable in practice.

In addition, they can be either integrated in a centralized entity capable of computing intra-domain routing strategies that optimize the network load like a Path Computation Element (PCE) or SDN controller, or used as an off-line tool by the decision-makers for network planning operations.

The remainder of the paper is organized as follows. Section II is devoted to preliminaries and basic definitions. In Section III, we introduce some necessary notations and give a formal definition of the problem along with a compact MILP formulation. We further present two families of inequalities valid for the problem. We describe our MILP-based iterative solving approach in Section V and a dynamic programming algorithm in Section VI. Finally, Section VII is devoted to present some early experiments to show the efficiency of our algorithms for state-of-the-art instances while some concluding remarks are given in Section VIII.

II. Preliminaries

In this section, we give the graph notations and notions used throughout this paper

a) Graph terminology: Let $G = (V, A)$ be a (un)directed graph. We also use the notation $V(G)$ and $A(G)$ (resp. $E(G)$) to denote the vertex set and arcs set (resp. edges set) of G respectively Let $X \subseteq V$, we denote by $G[X]$ the subgraph of G induced by X. We denote by $N(v)$ the set of adjacent vertices of v and $\delta^+(v)$ (resp. $\delta^-(v)$) the set of outgoing (resp. ingoing) arcs of v. A *path* or *v_1v_ℓ-path* is a sequence of vertices $v_1 - v_2 - \ldots - v_\ell$ such that $v_iv_{i+1} \in A$ for each $i = 1, \ldots, \ell - 1$. If no vertices appear more than once in a path then it is *elementary*. The two vertices v_1 and v_ℓ are the *endpoints* of the path and the others are called *internal* vertices. A subpath is defined as any subsequence $v_j - v_{j+1} - \ldots - v_{j+k}$ for some j and k. Unless stated otherwise, all the paths considered in this paper are elementary. We use the notation $p[v_i, v_j]$ to denote a subpath of p with v_i, v_j being any two vertices of p. We denote by $\mathcal{P}(G)$ the set of all elementary path in G.

We say that G is *bidirected* if $uv \in A$ and $vu \in A$ for all $u, v \in V$. The *underlying undirected graph* G^u of G is the undirected graph obtained from G by taking the same set of vertices, and with the set of edges defined as follows. There is an edge between any pair of vertices u and v, if the directed graph has an arc uv or vu.

b) Tree decomposition and treewidth: A tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of an undirected graph $G = (V, E)$ consists of a tree $T = (X, F)$ with node set X and edge set F, and a set $\mathcal{B} \subseteq 2^V$ whose members $B_x \in \mathcal{B}$, called *bags*, are labeled with the node $x \in X$, such that the following conditions are met:

1) $\bigcup_{x \in X} B_x = V$.
2) For each $uv \in E$ there is an $x \in X$ with $u, v \in B_x$.
3) For each $v \in V$, the node set $\{x \in X : v \in B_x\}$ induces a subtree of T.

The third condition is equivalent to assuming that if $v \in B_{x'}$ and $v \in B_{x''}$ then $v \in B_x$ holds for every node x of the (unique) $x'x''$-path in T. The width of a tree decomposition $\mathcal{T}$ is $w(\mathcal{T}) = \max_{x \in X} |B_x| - 1$ and the treewidth of G is defined as $\mathrm{tw}(G) = \min_{\mathcal{T}} w(\mathcal{T})$ where the minimum is taken over all tree decompositions $\mathcal{T} = (T, \mathcal{B})$ of G. The "-1" in the definition of $w(\mathcal{T})$ is included for the convenience that trees have treewidth 1 (rather than 2).

Any tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of a graph can be transformed in linear time into a so-called nice tree decomposition $\mathcal{T}' = (T', \mathcal{B}')$ with $w(\mathcal{T}') = w(\mathcal{T})$, $|\mathcal{B}'| = O(|\mathcal{B}|)$ and with $B_x \neq \emptyset$ for all $B_x \in \mathcal{B}$ where T' is a rooted tree satisfying the following conditions (see [12] for more details):

1) Each node of T' has at most two children.
2) For each node x with two children y, z, we have $B'_y = B'_z = B'_x$ (x is called *join node*) with $B'_x, B'_y, B'_z \in \mathcal{B}'$.
3) If a node x has just one child y, then $B'_x \subset B'_y$ (x is called *forget node*) or $B'_y \subset B'_x$ (x is called *insert node*) and $||B'_x| - |B'_y|| = 1$ with $B'_x, B'_y \in \mathcal{B}'$.

One can see that the subtree T_x of T rooted at node x represents the subgraph G_x induced by precisely those vertices of G which occur in at least one B_y where y runs over the nodes of T_x. When the graph is directed, the tree decomposition applies for the underlying undirected graph.

c) Parameterized complexity: The parameterized complexity theory is a framework that provides a new way to express the computational complexity of optimization problems. We briefly recall here the main ideas behind this theory, the reader is referred to [8] for more background on this subject. A problem parameterized by k is called *fixed-parameter tractable* (fpt) if there exists an algorithm, called an fpt algorithm, that solves it in time $f(k) \cdot n^{O(1)}$ (fpt-time) where n is the size of the input. The function f is typically super-polynomial and only depends on k. In other words, the combinatorial explosion is confined into f. If the values of k are small in practice, then the algorithm adopts a polynomial behavior.

III. Description of the problem

A. Notations and definitions

In terms of graphs, the problem can be presented as follows. We consider given a bidirected graph $G = (V, A)$ that represents an IP network topology. Every node $v \in V$ corresponds to a router while an arc $a = uv \in A$ represents a logical link between router nodes u and v. Every arc uv is associated a capacity (bandwidth) denoted by $c_{uv} \geqslant 0$ and a latency value denoted $\delta_{uv} \geqslant 0$. We let K denote a set of commodities (traffic demands) to be routed over the graph G. Every commodity k is defined by a pair (s^k, t^k) with s^k, t^k being the origin and destination of k, respectively, along with the traffic volume $D^k \geqslant 0$ to be routed from s^t to t^k and a a maximum delay value $\Delta^k \geqslant 0$.

Definition 1 (Partial routing (sub)path): A *partial routing path* for a commodity $k \in K$ in a graph G is a pair (p, k) denoted by p^k where p is a path. A *partial routing subpath* of p^k is a routing path q^k such that q is a subpath of p.

Definition 2 (Complete routing path): A routing path p^k is said to be complete for a commodity $k \in K$ in a graph G if the endpoints of p are exactly the origin and destination of k.

Definition 3 (Routing configuration): A *routing configuration* for a set of commodities K in a graph G is a subset $R_{G,K} \subseteq \{p^k : p \in \mathcal{P}(G), k \in K\}$ of routing paths.

The set of all possible routing configurations of K in G is denoted by $\mathcal{R}(G, K)$.

Definition 4 (Feasible routing configuration): A routing configuration $R \in \mathcal{R}(G, K)$ is said to be *feasible* if there exists a *weight function* $w : A \to \mathbb{Z}_+$ such that (i) for each $p^k \in R$ the path p is the unique shortest path between its endpoints according to w and (ii) the delay constraint is satisfied *i.e* $\sum_{p^k \in R} \sum_{uv \in p} \delta_{uv} \leqslant \Delta^k$.

Definition 5 (Complete routing configuration): A routing configuration $R \in \mathcal{R}(G, K)$ is said to be *complete* if it is feasible and there exists a (necessarily unique) complete routing path in R for each $k \in K$.

Definition 6 (Conflicting paths): Two paths p_1 and p_2 are said to be *conflicting* if they share two vertices u and v and $p_1[u, v] \neq p_2[u, v]$ with $p_1[u, v] \neq \emptyset$ and $p_2[u, v] \neq \emptyset$. Otherwise, they are said to satisfy Bellman property.

Finally, we provide the following two metrics:

Definition 7 (Load): The *load* $\texttt{load}(R, u, v)$ of an arc $uv \in A$ given a routing configuration $R \in \mathcal{R}(G, K)$ is defined as

$$\texttt{load}(R, u, v) = \frac{1}{c_{uv}} \sum_{p^k \in R : uv \in p} D^k$$

The load is the ratio between the total flow that goes through an arc and the arc's capacity.

Definition 8 (Congestion): The *congestion* $\texttt{cong}(R)$ of a routing configuration $R \in \mathcal{R}(G, K)$ is defined as

$$\texttt{cong}(R) = \max_{uv \in A} \texttt{load}(R, u, v)$$

The congestion is then the maximum load over all arcs.

B. Properties

In this section, we will state two lemmas that will be useful in the rest of the paper.

Lemma 1: Let $R \in \mathcal{R}(G, K)$ if R is feasible then it contains no conflicting routing paths.

Proof: Suppose, on the contrary, that R is feasible and yet contains two routing paths $p_1^k, p_2^{k'}$ such that they share two vertices u and v and $p_1[u, v] \neq p_2[u, v]$ with $p_1[u, v] \neq \emptyset$ and $p_2[u, v] \neq \emptyset$. Since R is feasible then there exists a weight function such that p_1 (resp. p_2) is the unique shortest path between its endpoints with respect to w. By the Bellman principle, the subpath $p_1[u, v]$ is a shortest between u and v. For the same reason, the subpath $p_2[u, v]$ is a shortest between u and v which is different from $p_1[u, v]$ by assumption. Hence, there are two different shortest paths to join the endpoints, denoted s and t, of p_1 i.e the path p_1 itself and the one formed by the subpaths $p_1[s, u]$, $p_2[u, v]$ and $p_1[v, t]$. This contradicts the unicity of p_1. ∎

The next lemma shows that feasibility can checked in polynomial time.

Lemma 2 (Benameur and Gourdin [1]): Determining whether a routing configuration is feasible and returning the corresponding weight function, if any, can be done in polynomial time.

C. Problem statement

We are now in position to state the problem studied in this paper. The *Delay Constrained Minimum Congestion* (D-USPR) problem is to find a set of weights to assign to the arcs of G and a set of routing paths induced by those weights such that (i) there is a unique shortest path satisfying the delay constraints for each commodity according to the identified weights and (ii) the network congestion is minimum. Formely, the problem is defined as follows.

D-USPR
Input: A bidirected graph $G = (V, A, c, \delta)$ where each arc $uv \in A$ has a *capacity* value $c_{uv} \geqslant 0$ and a *latency* value $\delta_{uv} \geqslant 0$, a set K of commodities where each commodity $k \in K$ is defined as (s^k, t^k, D^k, Δ^k).
Output: A complete routing configuration $R_{G,K}$ with minimum congestion value.

In this paper, we will also make use of this slightly more general version of the above problem

PRE ROUTED D-USPR
Input: A bidirected graph $G = (V, A, c, \delta)$ where each arc $uv \in A$ has a *capacity* value $c_{uv} \geqslant 0$ and a *latency* value $\delta_{uv} \geqslant 0$, a set K of commodities partitioned into two sets K_{free} (free demands) and K_{fixed} (fixed demands) and a complete routing configuration $R_{G,K_{fixed}}$.
Output: A complete routing configuration $R_{G,K_{free}}$ such that $R_{K_{free}} \cup R_{K_{fixed}}$ is feasible and has minimum congestion value.

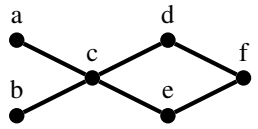
Observe that if $K_{fixed} = \emptyset$, we end up with the D-USPR problem.

It is worth noting that, even if they look similar at first glance, the EDGE DISJOINT PATHS (edp) problem (resp. NODE DISJOINT PATHS (ndp) problem) is not a particular case of D-USPR with unit demands and unit capacities (see Fig. 1). Recall that the edp (resp. ndp) problem asks, given an undirected graph and a set of k demands, to find k edge-disjoint (resp. node-disjoint) paths joining the demands. Consequently, the negative and positive results for edp or ndp do not directly transfer to D-USPR.

IV. MILP FORMULATION

A. Notations and formulation

Let x_a^k be a binary variable that takes the value 1 if commodity k is routed along a path using arc a and 0



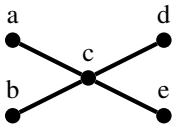


Fig. 1. In the graph on the left, the two paths $a-c-d-f$ and $b-c-e-f$ form the unique valid solution for the EDGE DISJOINT PATHS problem where the demands are (b,f) and (a,f). However, this is not a feasible routing configuration since these two paths are conflicting. In the graph on the right, it is not possible to find two node-disjoint paths satisfying the demands (b,e) and (a,d), yet it is easy to see that the routing paths $a-c-d$ and $b-c-e$ form a feasible routing configuration.

otherwise. We define the binary variables u_a^t that takes the value 1 if a belongs to a shortest path towards destination t and 0 otherwise. We further let w_{uv} denote the weight assigned to the arc uv and r_u^v be the potential of node u, that is the distance between node u and node v. The D-USPR problem is then equivalent to the following MILP formulation:

$$\min L \tag{1}$$

$$s.t. \sum_{a\in\delta^+(v)} x_a^k - \sum_{a\in\delta^-(v)} x_a^k = \begin{cases} 1 & \text{if } v = s^k, \\ -1 & \text{if } v = t^k, \\ 0 & \text{otherwise.} \end{cases} \quad \begin{matrix} \forall v \in V, \\ \forall k \in K, \end{matrix} \tag{2}$$

$$\sum_{k\in K} D^k x_a^k \leqslant c_{uv} L, \forall a \in A, \tag{3}$$

$$\sum_{a\in A} \delta_a x_a^k \leqslant \Delta^k, \forall k \in K, \tag{4}$$

$$\sum_{a\in\delta^+(v)} u_a^t \leqslant 1, \forall v \in V, \forall t \in T, \tag{5}$$

$$x_a^k \leqslant u_a^{t^k}, \forall a \in A, \forall k \in K, \tag{6}$$

$$u_a^t \leqslant \sum_{k\in K, t^k = t} x_a^k, \forall a \in A, \forall t \in T, \tag{7}$$

$$w_{uv} - r_u^t + r_v^t \geqslant 1 - u_{uv}^t, \forall uv \in A, \forall t \in T, \tag{8}$$

$$w_{uv} - r_u^t + r_v^t \leqslant M(1 - u_{uv}^t), \forall uv \in A, \forall t \in T, \tag{9}$$

$$x_a^k \in \{0,1\}, \forall k \in K, \forall a \in A, \tag{10}$$

$$u_a^t \in \{0,1\}, \forall a \in A, \forall t \in T, \tag{11}$$

$$w_{uv} \geqslant 0, \forall uv \in A, \tag{12}$$

$$r_u^v \geqslant 0, \forall u, v \in V \times V. \tag{13}$$

The objective (1) is to minimize the load of the most loaded link, denoted L. Inequalities (2) ensure that a unique path is associated to each commodity k and (3) express the load over an arc a. Inequalities (4) are the delay constraints over the routing paths while (5) and (6)-(7) are anti-arborescence and linking constraints, respectively. In particular, inequalities (5) ensure that there is at most one path traversing any node v towards a given destination $t \in T$, which is necessarily implied by Bellman property. Constraints (8) and (9) guarantee that the weight of any arc used by a shortest path towards a destination t corresponds to the difference of potentials between the end nodes of this arc and larger otherwise. Finally, (10)-(13) are the trivial and integrity constraints.

Proposition 1: The formulation (2)-(11) is valid for the D-USPR problem.

Proof: It is easy to see that any solution of D-USPR satisfies inequalities (2)-(11). To show the converse, let (x,u,w,r) denote a solution of (2)-(11) and consider two sets, say Q^k and S^t defined as follows: $Q^k = \{a \in A : x_a^k = 1\}$, for each $k \in K$, with $Q = \cup^{k\in K} Q^k$ and $S^t = \{a \in A : u_a^t = 1\}$, for every $t \in T$. Since $x \in \{0,1\}^{K\times A}$ satisfies the flow conservation constraints (2), Q^k clearly contains a unique routing $s^k t^k$-path for commodity k, which by (4) also satisfies the delay constraints. Inequalities (2), (5), (6) and (7) ensure that the elements of S^t form an anti-arborescence rooted at destination t and consequently, contains paths that satisfy Bellman property. Moreover, since $w \in \mathbb{R}^+$ and $r \in \mathbb{R}^+$ satisfy inequalities (8) and (9), every set Q^k, $k \in K$, consists of arcs which are tight with respect to the weights w, thus containing a shortest $s^k t^k$-path.

Now suppose that there are two conflicting paths in Q, say p^1 and p^2 and let s^1t^1 and s^2t^2 be their respective endpoints. Denote by u and v two internal nodes of p^1 and p^2 such that $p^1[u,v] \neq p^2[u,v]$. Further assume that

$$\sum_{v_i v_j \in p^1} w_{v_i v_j} = \sum_{u_i u_j \in p^2} w_{u_i u_j},$$

then the inequalities (8) and (9) with $t \in \{t_1, t_2\}$ summed over $v_i v_j \in p^1[u,v]$ and $u_i u_j \in p^2[u,v]$ yields a contradiction. Consequently, every st-path used for routing is a unique shortest path according to the weights w and hence (x,u,w,r) is a solution of the D-USPR problem. ■

B. Valid inequalities

In what follows, we present two families of inequalities valid for the D-USPR problem.

(i) Subpath consistency constraints

The first family is the so-called *subpath consistency constraints* and has been introduced in different versions in []

Proposition 2 ([4]): The following inequalities

$$x_a^{s,v} - x_a^{s,t} + \sum_{e\in\delta^-(v)} x_e^{s,t} \leqslant 1, \quad \forall (s,t),(s,v) \in K, \forall a \in A, \tag{14}$$

$$x_a^{v,t} - x_a^{s,t} + \sum_{e\in\delta^-(v)} x_e^{s,t} \leqslant 1, \quad \forall (s,t),(v,t) \in K, \forall a \in A, \tag{15}$$

are valid for the D-USPR problem.

Those inequalities ensure that two paths p^1, p^2 with a common endpoint $v_i \in V$, that intersect at a second node v_j are necessarily such that $p^1[v_i,v_j] = p^2[v_i,v_j]$. In other words, any pair of commodities having the same origin (respectively destination) node are necessarily routed along two paths that satisfy Bellman property.

(ii) Node precedence constraints

The second family of inequalities has been introduced by Garcia [10] for the *resource constrained shortest path* problem and later extended by Horvath et al. [11]. They arise directly from the maximum delay requirement of the D-USPR problem and express the fact that any feasible routing st-path using an arc $a = uv \in A$ should leave this arc through an arc $a' = u'v' \in \delta^+(v)$ satisfying the following condition

$$\sigma_{s,u} + \delta_a + \delta_{a'} + \sigma_{v',t} \leqslant \Delta^{s,t},$$

where $\sigma_{s,u}$ (respectively $\sigma_{v',t}$) is the length of the shortest path between nodes s and u (respectively between nodes v' and t) with respect to the delay metric. In other words, $\sigma_{uv} = \sum_{e \in p^{uv}} \delta_e$, for $(u, v) \in V \times V$. For each arc $a = uv \in A$ and each commodity $k \in K$ originating from node s^k and going to node t^k, denote by $\Phi^{out}_{a,k}$, the set of arcs defined as follows

$$\Phi^{out}_{a,k} = \{a' = u'v' \in \delta^+(v) \text{ with } v' \neq u : \sigma_{s^t,u} + \delta_a + \delta_{a'} + \sigma_{v',t^k} \leqslant \Delta^k\}.$$

Similarly, we let $\Phi^{in}_{a,k}$ denote the set of arcs entering into node u that can belong to a feasible routing path:

$$\Phi^{in}_{a,k} = \{a' = u'v' \in \delta^+(u) \text{ with } u' \neq v : \sigma_{s^t,u'} + \delta_{a'} + \delta_a + \sigma_{v,t^k} \leqslant \Delta^k\}.$$

Proposition 3: The following inequalities

$$x^k_a \leqslant \sum_{a' \in \Phi^{out}_{a,k}} x^k_{a,k}, \forall a \in A, \forall k \in K, \tag{16}$$

$$x^k_a \leqslant \sum_{a' \in \Phi^{in}_{a,k}} x^k_{a,k}, \forall a \in A, \forall k \in K, \tag{17}$$

are valid for the D-USPR problem.

Proof: Let k be a commodity of K with origin s^k and destination t^k and let $a = uv$ be an arc of A such that $v \neq s^k, t^k$. Denote by $(\overline{x}, \overline{u}, \overline{w}, \overline{r}, \overline{L})$ and let $p^k = \{e \in A : \overline{x}^k_e = 1\}$ be the $s^k t^k$-path associated with the routing of commodity k. It is easy to see that inequality (16) is trivially satisfied if $\overline{x}^k_a = 0$. Now if $\overline{x}^k_a = 1$, then, by (2), there exists an arc, say $a' = u'v'$ in $\delta^+(v)$ such that $\sum_{e \in \delta^+(v)} \overline{x}^k_a = 1$. Denote by $p[s^k u] = \{e \in A : \overline{x}^{s^k u}_e = 1\}$ (respectively $p[v't^k] = \{e \in A : \overline{x}^{v't^k}_e = 1\}$) the subpath with endpoints s^k, u (respectively, v', t^k. Suppose that a' is in $\delta^+(v) \setminus \Phi^{out}_{a,k}$, that is to say,

$$\sum_{e \in p[s^k,u]} \delta_e + \delta_a + \delta_{a'} + \sum_{e \in p[v',t^k]} \delta_e \geqslant$$

$$\sigma_{s^k,u} + \delta_a + \delta_{a'} + \sigma_{v',t^k} > \Delta^k,$$

which, by (4), yields a contradiction. Thus (16) are valid for D-USPR problem. We use similar arguments to show that inequalities (17) are valid for the problem. ■

V. An effective iterative algorithm

In this section, we introduce an effective iterative algorithm for solving the D-USPR problem. Consider given a graph G with a capacity vector C and a set of demands K with a latency vector Δ. The idea of this algorithm is to iteratively decrease the load by constructing feasible routing configurations and the associated weight vectors. To this end, we perform the following initialization steps:

Step 1 We first solve the *minimum congestion spanning tree* with delay constraints in G. Let H denote the spanning tree obtained.

Step 2 We set an arbitrary weight value w^0_{uv} on each arc uv of $A(H)$ and a infinite weight on the remaining arcs $uv \in A(G) \setminus A(H)$.

Step 3 We associate with each commodity of K a path, say p^k, in H between s^k and t^k.

We will denote by R_0 the complete routing configuration obtained at the end of the initialization phase. Note that R_0 obviously defines a feasible solution for the D-USPR problem.

We let $a^* \in A(G)$ be the most loaded arc with respect to R_0, that is to say $a^* = \arg\max_{uv \in A} \texttt{load}(R, u, v)$. At each iteration, we then apply the following procedure. We consider a partition of the demands set K into two subsets K_{fixed} and K_{free}, where K_{free} is the subset of (*congesting*) demands whose routing path in R_0 uses the arc a^* and K_{fixed} contains the remaining demands. We fix the routing paths of the demands in K_{fixed} along with the associated weights. Let T_{fixed} (resp. T_{free}) denote the destination nodes of the demands in K_{fixed} (resp. K_{free}) and $R^t \subseteq R_{K_{fixed}}$ denotes this fixed routing toward destination $t \in T_{fixed}$. We determine a feasible routing configuration by rerouting the demands of K_{free}, all other demands remaining equal. This can be done by solving the formulation (1)-(13) with the following changes. Inequalities (2), (4)-(9) are written over K_{free} and T_{free} instead of K and T while (3) is replaced by the following inequality

$$\sum_{k \in K_{free}} D^k x^k_a + \sum_{k \in K_{fixed}} D^k \overline{x}^k_a \leqslant c_{uv} L, \forall a \in A, \tag{18}$$

Finally, we add the following inequalities

$$w_{uv} - r^t_u + r^t_v = 0, \quad \forall uv \in R^t, \forall t \in T_{fixed}, \tag{19}$$

$$w_{uv} - r^t_u + r^t_v \geqslant 1, \quad \forall uv \in A \setminus R^t, \forall t \in T_{fixed}, \tag{20}$$

to ensure that the paths fixed in each set R^t define shortest paths towards the destination $t \in T_{fixed}$. This procedure is summarized in Algorithm 1.

VI. A dynamic programming algorithm

In this section, we introduce a dynamic programming algorithm based on a *tree decomposition* for solving the Pre Routed D-USPR problem. Observe that the problem is trivial in the case where the input graph is a tree since there can only be one path to route any demand. However, it is not possible to generalize this positive result to graphs of bounded treewidth since the problem is NP-complete even on bidirected

Algorithm 1: Iterative algorithm

Data: An instance (G, K, C, Δ) of the problem
Result: A complete routing configuration R
Initialization: $R^0 \leftarrow$ *a complete routing configuration obtained by performing Step1-Step3*
$a^* \leftarrow \arg\max_{a \in A} cong(R^0, a)$
$iter \leftarrow 0$
while $iter < iter_{max}$ **do**
  find the demands in K_{free} and K_{fixed}
  find a complete routing configuration $R^{iter}_{G,K_{fixed}}$
  $R_{iter} \leftarrow$ complete routing configuration obtained by solving (1)- (20)
  $a^* \leftarrow \arg\max_{a \in A} \texttt{load}(R_{iter}, a)$
  iter $\leftarrow$ iter + 1
return R_{iter} with $\texttt{cong}(R) \leqslant \texttt{cong}(R_0)$;

rings [3]. This negative result rules out the possibility of having an fpt algorithm parameterized only by the "treewidth". However, we prove in what follows that the problem is fixed-parameter tractable for the combined parameter "treewidth" and "number of demands".

Proposition 4: Given a nice tree decomposition of G^u of width ω, the PRE ROUTED D-USPR problem can be solved in at most $2^{|K|\omega^8+\Delta \log |K|} \cdot n^{O(1)}$-time where $\Delta = \max_{k \in K} \Delta^k$.

Proof: Let $I = (G = (V, A, c, \delta), K_{free}, K_{fixed}, R_{K_{fixed}})$ be an instance of PRE ROUTED D-USPR . Let $\mathcal{T} = (T = (X, F), \mathcal{B})$ be a nice tree decomposition of G^u rooted at node $r \in X$. We denote by ω the width of $\mathcal{T}$ and by n the order of G i.e $n = |V|$. We start the proof by introducing some extra notations and definitions.

Notations. Recall that T_x is the subtree of T rooted at node x and $G_x = (V_x, A_x)$ is the subgraph of G induced by the vertices of G which occur in at least one bag B_y where y runs over the nodes of T_x. In this proof, we will also use the subgraph $\bar{G}_x$ which is obtained from G_x by removing the arcs with both endpoints in B_x. We denote by U_x the set of all origins and destinations of the demands in K_{free} that lie in V_x i.e $U_x = \{\{s^k, t^k\} \cap V_x : k \in K_{free}\}$ (see Figure 2). Let $R \in \mathcal{R}(G, K)$ and G' a subgraph of G, we denote by $R|_{G'}$ the routing configuration obtained by taking the subpaths of R induced by G'. We denote by G_R the graph obtained by taking the union of all routing paths in R.

Definitions. In this paragraph, we introduce several notions that are needed in the proof.

Valid routing configuration: We say that a routing configuration $R \in \mathcal{R}(\bar{G}_x, K)$ is *valid*, if it is feasible and for every $k \in K$ one of the following two conditions is met:

- There is exacly one complete routing subpath $p^k \in R$.
- The graph induced by the union of the routing subpaths for k in R is made of disjoint paths with at least one endpoint in B_x. Furthermore, there are at most two degree-one vertices in $V_x \setminus B_x$ in such graph.

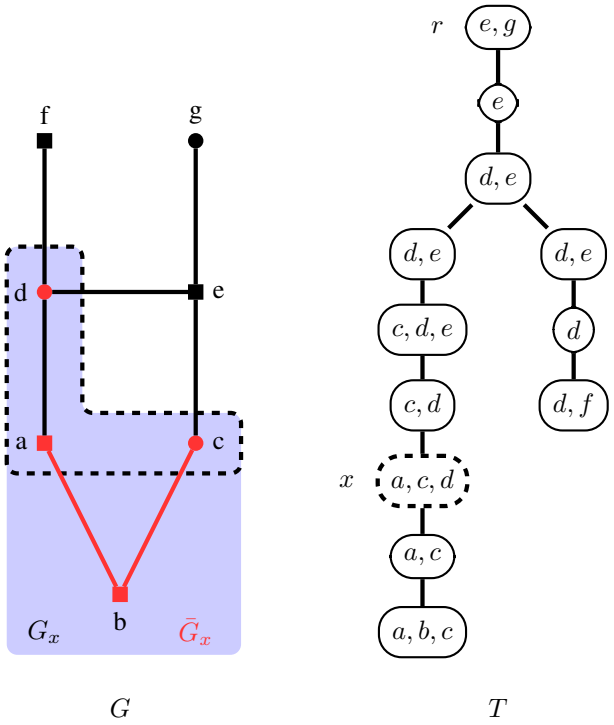


Fig. 2. Example of a graph G together with a nice tree decomposition $\mathcal{T} = (T = (X, F), \mathcal{H})$ of G^u rooted at node $r \in X$. In order to alleviate the figure and since the graph is bidirected, we drop the arcs orientation. We have $B_x = \{a, c, d\}$ and $U_x = \{a, b\}$. The origins and destinations of demands are represented with squares.

Routing contract: A "routing contract" H induced by a valid routing configuration $R \in \mathcal{R}(\bar{G}_x, K)$ is an edge-labeled graph where the edge labelling is a function λ_H from $E(H)$ to 2^K defined as follows. First, we say that a vertex $v \in V(G_R)$ is a *transit* vertex if it has degree 2 in G_R and the demands routed along the edges vv_1 and vv_2 according to R where $v_1, v_2 \in N(v)$ are the same. The graph H is then obtained from G_R by removing every transit vertex $v \in V(G_R)$ and inserting the edge v_1v_2 i.e we remove v and connect its neighbors with an edge. Regarding the edge labelling function λ_H, the demands in $\lambda_H(uv)$ are exactly those routed along the corresponding subpath $p[u, v]$ (which can be a single edge) in G_R (see Figure 3). More generally, we say that a routing configuration $Q \in \mathcal{R}(\bar{G}_x, K)$ is H-respecting if there exists a mapping $f : V(H) \to V(G_Q)$ such that for all $uv \in E(H)$ the demands in $\lambda_H(uv)$ are exactly those routed along the corresponding subpath $p[f(u), f(v)]$ in G_Q. We denote by $\mathcal{H}_x$ the set of all possible routing contracts at node x.

Delay contract: A "delay contract" induced by a valid routing configuration $R \in \mathcal{R}(\bar{G}_x, K)$ is a function $d : K \to \mathbb{N}$ defined as follows. Given a demand $k \in K$, the value $d(k)$ is equal to the sum of the delays on the arcs used to route the demand k according to R. We say that a routing configuration $Q \in \mathcal{R}(\bar{G}_x, K)$ is d-respecting if for each $k \in K$ we have $\sum_{p^k \in Q} \sum_{uv \in p} \delta_{uv} = d(k)$. We denote by $\mathcal{D}_x$ the set

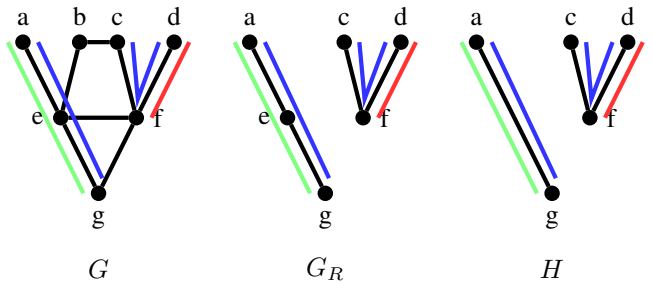


Fig. 3. Illustration of the construction of a routing contract. In this example, there are three demands k_1 (green), k_2 (blue) and k_3 (red) routed according to a valid routing configuration R. The different routing paths are represented as continuous colored line. The graph G_R is defined as the union of all routing paths in R. The routing contract H of R is then obtained by replacing the only transit vertex $e \in V(G_R)$ by the edge ag. Regarding the labeling function λ_H of H, it is defined as follows: $\lambda_H(ag) = \{k_1, k_2\}$, $\lambda_H(cf) = \{k_2\}$ and $\lambda_H(df) = \{k_2, k_3\}$.

of all possible delay contracts at node x.

Subproblems definition. We define a set of subproblems for each node $x \in X$, one corresponding to each possible $H_x \in \mathcal{H}_x$ and each $d_x \in \mathcal{D}_x$ that may represent in $\bar{G}_x$ the routing contract and delay contract induced by an optimal complete routing configuration for I. Hence, for each routing contract H_x and each delay contract d_x, we let $OPT_x(H_x, d_x)$ be an H_x-respecting and d_x-respecting valid routing configuration in $\mathcal{R}(\bar{G}_x, K)$ with minimum congestion. If no such routing configuration exists, we simply set $OPT_x(H_x, d_x) = \emptyset$.

Recurrence relations. We now describe how the solutions of the subproblems attached to a node are constructed. At the cost of adding more nodes in the tree T, we may assume w.l.o.g that the bags associated to the leaves of T contains only one vertex. In this case, each leaf is considered as an insert node. Initially, $OPT_x(H_x, d_x) = \emptyset$ for all $H_x, d_x \in \mathcal{H}_x \times \mathcal{D}_x$ and $x \in X$. By convention, $\texttt{cong}(\emptyset) = +\infty$. The algorithm computes the table OPT_x of each node x in T according to their type (insert, forget or join) and using a bottom-up procedure that ends to the root as follows.

a) Insert node: Let x be an insert node. In the case that x is a leaf, we simply skip this step and move on to the next node. Otherwise, let y be the child of x. By definition $B_y \subset B_x$ and $B_x \setminus B_y = \{v\}$. We compute the table OPT_x as follows. For each $H_y, d_y \in \mathcal{H}_y \times \mathcal{D}_y$ we perform the following instructions in sequence

- We define a routing contract H_x obtained from H_y by simply adding the vertex v.
- We construct a delay contract d_x by simply setting $d_x = d_y$.

After the instructions are performed, we set $OPT_x(H_x, d_x) = OPT_y(H_y, d_y)$.

b) Forget node: Let x be a forget node with child y. By definition $B_x \subset B_y$ and $B_y \setminus B_x = \{v\}$. Let E_v be the set of edges incident to v and $d_{B_x}(v) = |N(v) \cap B_x|$. This step requires the most attention since it is during this phase that we need to take care of the different ways to extend the routing paths of every routing configuration stored in OPT_y (See Figure 4). In what follows, we assume that whenever some fixed demands are routed along some of the edges in E_v we include the corresponding routing subpaths of $R_{K_{fixed}}$ into every routing extension constructed hereafter.

For each routing contract $H_y \in \mathcal{H}_y$ and each delay contract $d_y \in \mathcal{D}_y$ such that $OPT_y(H_y, d_y) \neq \emptyset$, we partition the set K_{free} into the following three sets:

- K_{open} : the free demands with no routing path in $OPT_y(H_y, d_y)$.
- $K_{partial}$: the free demands which have at least one (non-complete) routing path in $OPT_y(H_y, d_y)$.
- K_{closed} : the free demands for which there exists a complete routing path in $OPT_y(H_y, d_y)$.

First, we can ignore the demands in K_{closed} : since they are end-to-end routed, there are no more decisions to be made for them. Consider instead a free demand $k \in K_{open}$. Suppose for the moment that $v \neq s^k$ and $v \neq t^k$. Hence, this demand can be (possibly) routed through the vertex v using two edges of E_v. This yields to at most $d_{B_x}(v)(d_{B_x}(v)-1)/2$ choices to route k through v. Thus, a total of at most $(d_{B_x}(v)(d_{B_x}(v)-1)/2)^{|K_{open}|}$ possibilities to route all the demands in K_{open} in this way. If $v = s^k$ or $v = t^k$ then the only choice is to pick one of the edge in E_v to start (or finish) routing the demand. Clearly, the number of possibilities in this case is dominated by the previous case.

Now consider a free demand $k \in K_{partial}$. Let R_v^k be the set of routing paths for k in $OPT_y(H_y, d_y)$ having at least one endpoint in $(N(v) \cap B_x) \cup \{v\}$. We will show how many new routing paths can be obtained to route k through v by extending those in R_v^k. Similarly, suppose that $v \neq s^k$ and $v \neq t^k$. The demand k can be (possibly) routed through the vertex v by extending the paths in R_v^k with at most one or two edges of E_v. Thus the total number of possible ways to extend the routing paths in R_v^k is bounded by $(d_{B_x}(v)(d_{B_x}(v)-1)/2)$. Thus, a total of at most $(d_{B_x}(v)(d_{B_x}(v)-1)/2)^{|K_{partial}|}$ possibilities to route all the demands in $K_{partial}$. Suppose now that $v = s^k$ or $v = t^k$, if there exists a routing path $p^k \in R_v^k$ then the path cannot be extended through v. Otherwise, the only choice is to extend the routing paths in R_v^k by picking one of the edge in E_v to start (or finish) routing the demand. Clearly, the number of possibilities in this case is dominated by the previous case.

Overall, there are at most

$$(d_{B_x}(v)(d_{B_x}(v)-1)/2)^{|K_{closed}|+|K_{partial}|}$$

new possible routing configurations that can be constructed from the ones in $OPT_y(H_y, d_y)$. Let $\mathcal{R}_x$ be the set of those routing configurations that are valid (recall that checking whether a routing configuration is feasible can be done in polynomial time using Lemma 2). For each $R_x \in \mathcal{R}_x$, let $H_x \in \mathcal{H}_x$ and $d_x \in \mathcal{D}_x$ be the routing contract and delay contract induced by R_x (i.e R_x is H_x-respecting and d_x-respecting), we set $OPT_x(H_x, d_x) = R_x$ if $\texttt{cong}(R_x) < \texttt{cong}(OPT_x(H_x, d_x))$.

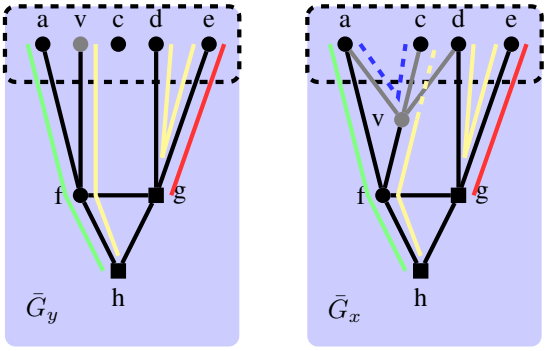


Fig. 4. Illustration of a possible routing configuration extension during a forget node operation. The edges in grey are those belonging to $E_v = \{va, vc, vd\}$ and we have $B_x = \{a, c, d, e\}$, $B_y = \{a, v, c, d, e\}$ and $d_{B_x}(v) = 3$. The different routing paths are represented as colored lines. Dashed lines corresponds to a possible extension of these routing paths.

c) Join node: Let x be a join node with two children y, z. By definition $B_y = B_z = B_x$. For each $H_y, d_y \in \mathcal{H}_y \times \mathcal{D}_y$ and each $H_z, d_z \in \mathcal{H}_z \times \mathcal{D}_z$, let $R_x = OPT_y(H_y, d_y) \cup OPT_z(H_z, d_z)$ and let $H_x, d_y \in \mathcal{H}_x \times \mathcal{D}_x$ be the routing contract and delay contract induced by R_x. We set $OPT_x(H_x, d_x) = R_x$ if R_x is valid and $\texttt{cong}(R_x) < \texttt{cong}(OPT_x(H_x, d_x))$.

d) Final step: Once we have computed the optimal solutions for every node, we can determine a complete routing configuration $R^* \in \mathcal{R}(G, K)$ with minimum congestion for the instance I as follows. Apply the forget node operation to every vertex in B_r to get a new table OPT, then return the solution $OPT(H, d)$ of minimum congestion among all H and d.

Correctness. The correctness follows from the following claim.

Claim 1: Let $x \in X$ and $R \in \mathcal{R}(\bar{G}_x, K)$ be a minimum congested valid routing configuration. For every child $y \in X$ of x, $OPT_y(H_y, d_y) \in \mathcal{R}(\bar{G}_y, K)$ is a minimum congested valid routing configuration where $H_y \in \mathcal{H}_y$ and $d_y \in \mathcal{D}_y$ are induced by $R|_{\bar{G}_y}$.

Proof: Let H and d be the routing contract and delay contract induced by R. Suppose that there exists a H_y-respecting and d_y-respecting valid routing configuration $R_y \in \mathcal{R}(\bar{G}_y, K)$ such that $\texttt{cong}(R_y) < \texttt{cong}(OPT_y(H_y, d_y))$. Consider the routing configuration $R' \in \mathcal{R}(\bar{G}_x, K)$ which is obtained by extending the routing paths of R_y the same way that $OPT_y(H_y, d_y)$ gets extended to obtain R. So R' is H-respecting, d-respecting and we have $\texttt{cong}(R') < \texttt{cong}(R)$. We claim that R' is a valid routing configuration which contradicts the choice of R as being a minimum congested valid routing configuration in $\mathcal{R}(\bar{G}_x, K)$. We show that R' is feasible since the other conditions of validity are satisfied by construction. Since R is feasible there exists a weight function w such that for each $p^k \in R$ the path p is the unique shortest path between its endpoints according to w. We show how to construct a weight function w' from w so that R' is feasible with respect to w'. For this purpose, we will use the fact that R and R' are both H-respecting. For each $uv \in E(H)$, there is corresponding routing subpath $p[u, v]$ in R and $p'[u, v]$ in R', and we set $w'(e') = \frac{1}{\ell} \sum_{e \in p[u,v]} w'(e)$ for each edge $e' \in p'[u, v]$ where ℓ is the length of $p[u, v]$. Finally, for every edge $uv \in E(\bar{G}_x)$ such that $w'(uv)$ is undefined, we set $w'(uv) = +\infty$. This finishes the construction of w' and we claim that R' is feasible according to w'. To see this, observe that any H-respecting routing graph is obtainable from H by subdividing an appropriate number of time each edge in $E(H)$. Thus, whenever there is a H-respecting routing graph that is feasible according to w, it suffices to construct a weight function w' that preserves the distances between the vertices of degree greater than two. The routing configuration R' is then valid and $\texttt{cong}(R') < \texttt{cong}(R)$ which contradicts the minimality of R. ∎

Running time. First, regarding the number of subproblems to solve, there are at most $|\mathcal{H}_x| \cdot |\mathcal{D}_x|$ of them associated to each node $x \in X$. This corresponds to the number of possible pair routing contract and delay contract at each node x. Since we have a nice tree decomposition of $O(n)$ nodes, we end up with a total of at most $O(|\mathcal{H}_x| \cdot |\mathcal{D}_x| \cdot n)$ subproblems to solve. The most costly subproblem to solve is the forget node operation which takes time at most

$$(d_{B_x}(v)(d_{B_x}(v) - 1)/2)^{|K_{closed}| + |K_{partial}|} \cdot n^{O(1)}$$

which is bounded by $\omega^{O(|K_{free}|)} \cdot n^{O(1)}$. Thus overall running time is

$$\omega^{O(|K_{free}|)} \cdot |\mathcal{H}_x| \cdot |\mathcal{D}_x| \cdot n^{O(1)}$$

Claim 2: Let $x \in X$, we have $|\mathcal{H}_x| \leq 2^{O(|K||B_x|^8)}$

Proof: By definition, $\mathcal{H}_x$ contains only routing contracts that are induced by valid routing configurations in $\mathcal{R}(\bar{G}_x, K)$. Let $H_x \in \mathcal{H}_x$ and $R \in \mathcal{R}(\bar{G}_x, K)$ be a valid H_x-respecting routing configuration. First, the number of routings paths in R is bounded by $O(|B_x|^2)$. Indeed, since R is valid, every $p^k \in R$ is either complete or has at least one endpoint in B_x. Hence, there can be as many routing subpaths as the number of pairs of vertices in B_x plus at most two routing paths per demand with exactly one endpoint in B_x. Indeed, if there are more routing paths then we may create conflicting paths which is ruled out by Lemma 1 or the graph induced by R may not contain only disjoint paths. Hence there can be no more than $|B_x|(|B_x| - 1)/2 + 2|B_x| = O(|B_x|^2)$ routing paths in R as claimed.

Now we will determine the maximum number of possible routing contracts that can be obtained from valid routing configurations in $\mathcal{R}(\bar{G}_x, K)$. Let p be a path in H_x that is used to route some demand in K. By definition of a routing contract, we know that every vertex in p intersects with at least one other routing path in R (recall that all transit vertices are removed). Moreover, since there is no conflicting paths in R (Lemma 1), every other routing path can intersect p at most once. Thus p^k has no more than $2|R| + 2$ vertices and then the graph H_x contains at most $|R|(2|R| + 2) = O(|B_x|^4)$ vertices. Finally, there can be at most $2^{|K||E(H_x)|}$ possible edge-labelling function for H_x. Hence, the number of possible

routing contract in $\mathcal{H}_x$ is bounded by $2^{O(|K||B_x|^8)}$ as claimed. ■

Claim 3: Let $x \in X$, we have $|\mathcal{D}_x| \leq |K|^{\Delta}$.

Proof: By definition, $\mathcal{D}_x$ contains only delay contracts that are induced by valid routing configurations in $\mathcal{R}(\bar{G}_x, K)$. Let $d_x \in \mathcal{D}_x$ and $R \in \mathcal{R}(\bar{G}_x, K)$ be a valid d_x-respecting routing configuration. Thus, for all $k \in K$, we have

$$d_x(k) = \sum_{p^k \in R} \sum_{uv \in p} \delta_{uv} \leqslant \Delta^k \leqslant \Delta$$

The number of possible delay contracts is then bounded by $|K|^{\Delta}$ as claimed. ■

Using Claim 2 and Claim 3, we deduce that the overall running time is bounded by $2^{|K|\omega^8+\Delta \log |K|}.n^{O(1)}$, as claimed. ■

Since finding an optimal tree-decomposition of a graph is fixed-parameter tractable with respect to the treewidth of that graph [7], we obtain the following result as an immediate corollary.

Proposition 5: The PRE ROUTED D-USPR problem is fixed-parameter tractable with respect to the parameter "treewidth" and "number of demands".

It is interesting to note that this algorithm can be compared with the two phases approach proposed in [4]. Indeed, the master problem that finds a set of routing paths is simply replaced here with a dynamic programming procedure while we still need the client to check for feasibility.

VII. NUMERICAL RESULTS

In this section, we present some early experiments related to the D-USPR problem and based on the results described above. Both exact and heuristic solving approaches are implemented in Python and using Cplex 12.8 with the default settings and NetworkX graph library. We have tested our algorithms with the following features:

- first by (i) solving the basic formulation (1)-(13),
- then by introducing (ii) the subpath consistency inequalities (14)-(15), (iii) the node-precedence inequalities (16)-(17) and (iv) both families of valid inequalities, to the basic formulation,
- Algorithm 1 uses a variant of the formulation (1)-(13), as described in Section V, along with both families of valid inequalities.

We have tested our algorithms on several instances derived from SNDlib[3] topologies of variying size and density. The big-M value is set to $|A| \times |K|$ for all the experiments, likewise in [4] and the CPU time limit is fixed to 5 hours for the exact approach.

Table I shows the impact of using valid inequalities and the efficiency of each class in strengthening the basic formulation (1)-(13) for nine instances. The first four columns refer to the name, number of nodes, number of arcs and number commodities, for each instance. Then, for each of the configurations (i) (basic MILP), (ii) (MILP + subpath consistency inequalities) and (iii) (MILP + node precedence inequalities), we show the following entries: Gap (%) is the root gap (the relative difference between the best upper bound (optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root node), Nodes is the number of nodes in the branch-and-bound tree and TT is the CPU time for computation (in seconds). The value in Gap column is written in italics if the solution found within the time limit was not optimal and replaced by "–" if no feasible solution was found within that time. We can see from Table I that the subpath consistency constraints (14)-(15) allow to improve the gap for several instances and help in reducing substantially both the number of nodes in the branch-and-bound tree and the CPU time for computation. In particular, except for *dfn-bwin* and *dfn-gwin*, that are the instances with highest density, all the instances are solved to optimality in less than 20 minutes, most of them at the root node. A less significant impact is obtained when using node precedence inequalities (16)-(17), yet they allow to speed up the resolution for several instances, and to reduce the size of the branch-and-bound tree (like for instance *di-yuan*).

Table II shows the results obtained when adding both families of inequalities to the basic formulation for the previous instances. We can see from the table that there is a positive but slight impact on the CPU time, especially for instances *Di-yuan* and *Nobel-US*.

Note that, these results although promising can be significantly improved by generating the valid inequalities in a dynamic fashion (via separation routines in a branch-and-cut framework) instead of being all integrated in the basic MILP.

We have tested our iterative algorithm on instances *France*, *Nobel-EU* and *Norway* that could not be solved using the exact approach due to their size, density and number of demands. Those three instances are among the most challenging state-of-the-art instances for the USPR problem. The iterative algorithm allows to obtain a good upper bound for all three instances simply by improving an existing solution (e.g. one obtained from the minimum spanning tree congestion). A preliminary set of results shows empirically that a good partition of the demands set with an appropriate selection of the demands to reroute (K_{free}) allows to substantially improve the trivial bound of the existing solution. In addition, the fact that we start from an existing solution allows us to save computation efforts highlighting even further the potential scalability of our algorithm. For example, in instance France, rerouting 3% of the demands allows to improve the minimum spanning tree congestion bound by 2% while an improvment of 7% is enabled by rerouting 16% of the demands.

VIII. CONCLUDING REMARKS

In this paper, we have investigated several research directions to go towards more scalable algorithmic solutions. For this purpose, we proposed the following two approches: (i) reducing the size of the problem (e.g number of demands) and (ii) exploiting the structure of the input graph (e.g treewidth). Although the obtained results are promising there is still room

[3]http://sndlib.zib.de

TABLE I
THE IMPACT OF EACH CLASS OF VALID INEQUALITIES

				(i) Basic MILP			(ii) MILP + subpath cons. ineq.			(iii) MILP + node precendence ineq.		
Topology	$\|V\|$	$\|A\|$	$\|K\|$	Gap (%)	Nodes	TT	Gap (%)	Nodes	TT	Gap (%)	Nodes	TT
PDH	11	68	24	0.00	1	0.80	0.00	1	0.41	0.00	1	0.34
Di-yuan	11	84	22	0.00	6571	773.41	0.00	1	1.24	0.00	1	0.45
Polska	12	36	66	6.70	602765	17302.36	6.62	71	34.66	6.62	78295	6780.74
Nobel-US	14	42	91	7.98	19004	18000.00	2.02	5	439.08	-	255813	18000.00
Dfn-bwin	10	90	90	0.00	1	0.45	51.25	1	18000.00	0.00	1	1.91
abilene	12	30	132	0.00	6832	19.49	0.00	1	8.59	0.00	6271	17.77
Dfn-gwin	11	94	110	6.67	5375	862.72	37.82	1	18000.00	12.03	31	707.97
Atlanta	15	44	210	0.00	10839	187.22	0.25	1	20.58	0.00	125393	6280.28
Nobel-GER	17	52	121	-	-	18000.00	12.12	1	811.39	-	385146	18000.00

TABLE II
THE IMPACT OF ADDING BOTH CLASSES OF INEQUALITIES

Name	LB	UB	root gap (%)	Nodes	TT (sec.)
PDH	12.80	12.80	0.00	1	0.50
Di-yuan	5.00	5.00	0.00	1	0.30
Polska	6.41	6.90	6.70	243	60.97
Nobel-US	24.20	24.70	2.02	5	263.86
Dfn-bwin	0.34	0.69	50.72	1	18000.00
abilene	60.41	60.4	0	1	10.17
Dfn-gwin	0.65	1.05	38.10	1	18000.00
Atlanta	3.58	3.58	0,00	5	29.64
Nobel-GER	3.87	4.40	12.12	16	458.00

for improvment. First, we expect that solving the formulation using a branch-and-cut algorithm will subtantially improve the performance of the MILP-based exact approach and the efficiency of the iterative algorithm. Second, we are implementing the dynamic programming algorithm and even though the theoretical complexity is prohibitive, we believe that the efficiency of this algorithm can be good in practice especially if used in combination with a parallelization approach. Finally, this problem seems to be a good candidate to apply machine learning methods in the hope to reach better running time through learned heuristics.

REFERENCES

[1] W. Ben-Ameur and É. Gourdin. Internet routing and related topology issues. *SIAM J. Discrete Math.*, 17(1):18–49, 2003.

[2] A. Bley. *Routing and capacity optimization for IP networks*. PhD thesis, Technische Univertität Berlin, 2007.

[3] A. Bley. Approximability of unsplittable shortest path routing problems. *Networks*, 54(1):23–46, 2009.

[4] A. Bley. An integer programming algorithm for routing optimization in IP networks. *Algorithmica*, 60(1):21–45, 2011.

[5] A. Bley, B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pióro, A. Tomaszewski, and H. Ümit. *Optimization of OSPF Routing in IP Networks*, pages 199–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] A. Bley, M. Grötschel, and R. Wessäly. Design of broadband virtual private networks: Model and heuristics for the b-win. In *Robust Communication Networks: Interconnection and Survivability*, 1998.

[7] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[8] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag, 2013.

[9] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications*, pages 519–528. IEEE Computer Society, 2000.

[10] R. Garcia. *Resource constrained shortest paths and extensions*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2009.

[11] M. Horváth and T. Kis. Solving resource constrained shortest path problems with lp-based methods. *Computers & Operations Research*, 73:150 – 164, 2016.

[12] T. Kloks. *Treewidth, Computations and Approximations*. Springer, 1994.

[13] A. Parmar, S. Ahmed, and J. Sokol. An integer programming approach to the ospf weight setting problem. 2006.

[14] N. Perrot, A. Benhamiche, Y. Carlinet, and E. Gourdin. *Future Networks: Overview of Optimization Problems in Decision-Making Procedures*, pages 177–207. IGI Global, 2019.