

# On Polynomial-Time Combinatorial Algorithms for Maximum $L$ -Bounded Flow

Kateřina Altmanová, Petr Kolman, Jan Voborník  
 Department of Applied Mathematics  
 Faculty of Mathematics and Physics  
 Charles University, Prague

February 21, 2019

## Abstract

Given a graph  $G = (V, E)$  with two distinguished vertices  $s, t \in V$  and an integer  $L$ , an  $L$ -bounded flow is a flow between  $s$  and  $t$  that can be decomposed into paths of length at most  $L$ . In the *maximum  $L$ -bounded flow problem* the task is to find a maximum  $L$ -bounded flow between a given pair of vertices in the input graph.

The problem can be solved in polynomial time using linear programming. However, as far as we know, no polynomial-time combinatorial algorithm for the  $L$ -bounded flow is known. The only attempt, that we are aware of, to describe a combinatorial algorithm for the maximum  $L$ -bounded flow problem was done by Koubek and Říha in 1981. Unfortunately, their paper contains substantial flaws and the algorithm does not work; in the first part of this paper, we describe these problems.

In the second part of this paper we describe a combinatorial algorithm based on the exponential length method that finds a  $(1 + \epsilon)$ -approximation of the maximum  $L$ -bounded flow in time  $O(\epsilon^{-2} m^2 L \log L)$  where  $m$  is the number of edges in the graph. Moreover, we show that this approach works even for the NP-hard generalization of the maximum  $L$ -bounded flow problem in which each edge has a length.

## 1 Introduction

Given a graph  $G = (V, E)$  with two distinguished vertices  $s, t \in V$  and an integer  $L$ , an  $L$ -bounded flow is a flow between  $s$  and  $t$  that can be decomposed into paths of length at most  $L$ . In the *maximum  $L$ -bounded flow problem* the task is to find a maximum  $L$ -bounded flow between a given pair of vertices in the input graph. The  $L$ -bounded flow was first studied, as far as we know, in 1971 by Adánek and Koubek [1]. In connection with telecommunication networks,  $L$ -bounded flows in networks with unit edge lengths have been widely studied and are known as *hop-constrained flows* [7].

For networks with unit edge lengths (or, more generally, with polynomially bounded edge lengths, with respect to the number of vertices), the problem can be solved in polynomial time using linear programming. Linear programming is a very general tool that does not make use of special properties of the problem at hand. This often leaves space for superior combinatorial algorithms that do exploit the structure of the problem. For example, maximum flow, matching, minimum spanning tree or shortest path problems can all be described as linear programs but there are many algorithms that outperform general linear programming approaches. However, as far as we know, no polynomial-time combinatorial algorithm<sup>1</sup> for the  $L$ -bounded flow is known.

---

<sup>1</sup>Combinatorial in the sense that it does not explicitly use linear programming methods or methods from linear algebra or convex geometry.

## 1.1 Related results

For clarity we review the definitions of a few more terms that are used in this paper. A *network* is a quintuple  $G = (X, R, c, s, t)$ , where  $G = (X, R)$  is a directed graph,  $X$  denotes the set of vertices,  $R$  the set of edges,  $c$  is the edge capacity function  $c : R \rightarrow \mathbb{R}^+$ ,  $s$  and  $t$  are two distinguished vertices called the source and the sink. We use  $m$  and  $n$  to denote the number of edges and the number of vertices, respectively, in the network  $G$ , that is,  $m = |R|$  and  $n = |X|$ . Given an  $L$ -bounded flow  $f$ , we denote by  $|f|$  the size of the flow, and for an edge  $e \in R$ , we denote by  $f(e)$  the total amount of flow  $f$  through the edge  $e$ .

An  *$L$ -bounded flow problem with edge lengths* is a generalization of the  $L$ -bounded flow problem: each edge has also an integer length and the length of a path is computed not with respect to the number of edges on it but with respect the sum of lengths of edges on it.

Given a network  $G$  and an integer parameter  $L$ , an  *$L$ -bounded cut* is a subset  $C$  of edges  $R$  in  $G$  such that there is no path from  $s$  to  $t$  of length at most  $L$  in the network  $G = (X, R \setminus C, c, s, t)$ . The objective is to find an  $L$ -bounded cut of minimum size. We sometimes abbreviate the phrase  $L$ -bounded cut to  $L$ -cut and, similarly, we abbreviate the phrase  $L$ -bounded flow to  $L$ -flow.

Although the problems of finding an  $L$ -flow and an  $L$ -cut are easy to define and they have been studied since the 1970's, still some fundamental open problems remain unsolved. Here we briefly survey the main known results.

**L-bounded flows** As far as we know, the  $L$ -bounded flow was first considered in 1971 by Adámek and Koubek [1]. They published a paper introducing the  $L$ -bounded flows and cuts and describing some interesting properties of them. Among other results, they show that, in contrast to the ordinary flows and cuts, the duality between the maximum  $L$ -flow and the minimum  $L$ -cut does not hold.

The maximum  $L$ -flow can be computed in polynomial time using linear programming [4, 17, 4, 21]. The only attempt, that we are aware of, to describe a combinatorial algorithm for the maximum  $L$ -bounded flow problem was done by Koubek and Říha in 1981 [18]. The authors say the algorithm finds a maximum  $L$ -flow in time  $O(m \cdot |I|^2 \cdot S / \psi(G))$ , where  $I$  denotes the set of paths in the constructed  $L$ -flow,  $S$  is the size of the maximum  $L$ -flow, and  $\psi(G) = \min(|c(e) - c(g)| : c(e) \neq c(g), e, g \in R \cup \{e'\})$ , where  $c(e') = 0$ . Unfortunately, their paper contains substantial flaws and the algorithm does not work as we show in the first part of this paper. Thus, it is a challenging problem to find a polynomial time combinatorial algorithm for the maximum  $L$ -bounded flow.

Surprisingly, the maximum  $L$ -bounded flow problem with edge lengths is NP-hard [4] even in outer-planar graphs. Baier [3] describes a FPTAS for the maximum  $L$ -bounded flow with edge lengths that is based on the ellipsoid algorithm. He also shows that the problem of finding a decomposition of a given  $L$ -bounded flow into paths of length at most  $L$  is NP-hard, again even if the graph is outer-planar.

A related problem is that of  $L$ -bounded disjoint paths: the task is to find the maximum number of vertex or edge disjoint paths, between a given pair of vertices, each of length at most  $L$ . The vertex version of the problem is known to be solvable in polynomial time for  $L \leq 4$  and NP-hard for  $L \geq 5$  [15], and the edge version is solvable in polynomial time for  $L \leq 5$  and NP-hard for  $L \geq 6$  [6].

**L-bounded cuts** The  $L$ -bounded cut problem is NP-hard [22]. Baier et al. [4] show that it is NP-hard to approximate it by a factor of 1.377 for  $L \geq 5$  in the case of the vertex  $L$ -cut, and for  $L \geq 4$  in the case of the edge  $L$ -cut. Assuming the Unique Games Conjecture, Lee et al. [19] proved that the minimum  $L$ -bounded cut problem is NP-hard to approximate within any

constant factor. For planar graphs, the problem is known to be NP-hard [10, 24], too.

The best approximations that we are aware of are by Baier et al. [4]: they describe an algorithm with an  $\mathcal{O}(\min\{L, n/L\}) \subseteq \mathcal{O}(\sqrt{n})$ -approximation for the  $L$ -bounded vertex cut, and  $\mathcal{O}(\min\{L, n^2/L^2, \sqrt{m}\}) \subseteq \mathcal{O}(n^{2/3})$ -approximation for the  $L$ -bounded edge cut. The approximation factors are closely related with the cut-flow gaps: there are instances where the minimum edge  $L$ -cut (vertex  $L$ -cut) is  $\Theta(n^{2/3})$ -times ( $\Theta(\sqrt{n})$ -times) bigger than the maximum  $L$ -flow [4]. For the vertex version of the problem, there is a  $\tau$ -approximation algorithm for graphs of treewidth  $\tau$  [16].

The  $L$ -bounded cut was also studied from the perspective of parameterized complexity. It is fixed parameter tractable (FPT) with respect to the treewidth of the underlying graph [8, 16]. Golovach and Thilikos [12] consider several parameterizations and show FPT-algorithms for many variants of the problem (directed/undirected graphs, edge/vertex cuts). On planar graphs, it is FPT with respect to the length bound  $L$  [16].

The  $L$ -bounded cut appears in the literature also as the short paths interdiction problem [5], [16], [19] or as the most vital edges for shortest paths [5].

## 1.2 Our contributions

In the first part of the paper, we show that the combinatorial algorithm by Koubek and Říha [18] for the maximum  $L$ -bounded flow is not correct.

In the second part of the paper we describe an iterative combinatorial algorithm, based on the exponential length method, that finds a  $(1 + \epsilon)$ -approximation of the maximum  $L$ -bounded flow in time  $\mathcal{O}(\epsilon^{-2} m^2 L \log L)$ ; that is, we describe a fully polynomial approximation scheme (FPTAS) for the problem.

Moreover, we show that this approach works even for the NP-hard generalization of the maximum  $L$ -bounded flow problem in which each edge has a length. This approach is more efficient than the FPTAS based on the ellipsoid method [3].

Our result is not surprising (e.g., Baier [3] mentions the possibility, without giving the details, to use the exponential length method to obtain a FPTAS for the problem); however, considering the absence of other polynomial time algorithms for the problem that are not based on the general LP algorithms, despite of the effort to find some, we regard it as a meaningful contribution. The paper is based on the results in the bachelor's thesis of Kateřina Altmanová [2] and in the master's thesis of Jan Voborník [23].

## 2 The algorithm of Koubek and Říha

### 2.1 Increasing an $L$ -bounded flow

Before describing the problem with the algorithm by Koubek and Říha [18], we informally describe the purpose and the main attributes of an *increasing  $L$ -system*, a key structure used in the algorithm.

Consider a network  $G = (X, R, c, s, t)$  and an arbitrary  $L$ -bounded flow  $f$  from  $s$  to  $t$  in  $G$ , together with its decomposition into paths of length at most  $L$  (say paths  $p_1, p_2, \dots$  carrying  $r_1, r_2, \dots$  units of flow, resp.) that is not a maximum  $L$ -bounded flow. Given  $G$  and  $f$ , Koubek and Říha [18] build a labeled oriented tree  $T = (V, E, v_0, LABV, LABE)$  where  $V$  is the set of nodes,  $E$  is the set of edges,  $v_0$  is the root,  $LABV$  is a vertex labelling and  $LABE$  is an edge labeling. The tree is called an *increasing  $L$  system with respect to  $f$* .

There are four types of the nodes of the tree  $T$ ; to explain the error in the paper, it is sufficient to deal with three of them: 1-son, 3-son, 4-son. With (almost) each node  $u$  in  $T$ , are associated two consecutive paths in  $G$ : the first one, denoted by  $q(u)$ , contains only edges that

are not used by the current  $L$ -flow  $f$ , and the second one, denoted by  $\bar{q}(u)$ , coincides with a subpath of some path from the current  $L$ -flow  $f$ . (Fig. 1). The tree  $T$  encodes a combination

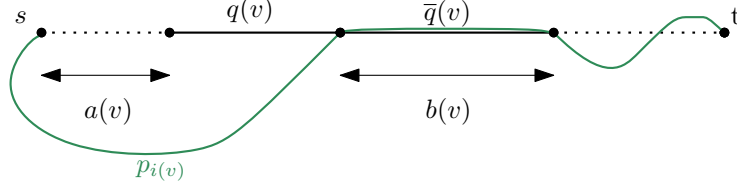


Figure 1: The concatenation of the section  $q(v)$  to  $\bar{q}(v)$ .

of these paths with paths in  $f$  and this combination is supposed to yield a larger  $L$ -flow than the  $L$ -flow  $f$ .

The label of a vertex  $v$  in the tree  $T$ , denoted by LABV in the original paper, and the label of the edge  $e$  connecting  $v$  to its immediate ancestor, if there is one, denoted by LABE, are of the following form:

	LABV	LABE
1-son	$(q(v), i(v), a(v), b(v))$	none
3-son or 4-son	$(q(v), i(v), a(v), b(v))$	$(h(e), j(e), d(e), o(e))$

where

- $q(v)$  is a path in  $G$  that is edge disjoint with every path in the  $L$ -flow  $f$ ,
- $i(v), j(v)$  are indices of paths in the  $L$ -flow  $f$ ,
- $a(v), b(v), d(e)$  are positive integers (distances),
- $o(e)$  is a positive integer, if  $v$  is a 3-son, and  $o(v)$  is a pointer to a 3-son, if  $v$  is a 4-son,
- $h(e)$  is a subset of edges in  $G$ .

As for every node  $v$  in the tree (except for the root) there is a unique edge  $e$  connecting it to its parent, Koubek and Říha often refer to the label of the edge  $e$ , and to its attributes, by the name of the vertex  $v$ , e.g., they write  $h(v)$  instead of  $h(e)$ ; we shall use the same convention.

The tree  $T$  is supposed to describe an  $L$ -flow  $f'$  derived from  $f$ . In particular, each path  $q(v)$  and  $\bar{q}(v)$  is a subpath of a new path between  $s$  and  $t$  of length at most  $L$ . Very roughly speaking, the attributes  $a(v)$  and  $d(v)$  store information about the distance of the path segments  $q(v)$  and  $\bar{q}(v)$  from  $s$  along the paths used in the new  $L$ -flow  $f'$ , the attribute  $i(v)$  specifies the index of a path from  $f$  s.t.  $\bar{q}(v)$  is a subpath of  $p_{i(v)}$ , and the attributes  $b(v)$  and  $o(v)$ , resp., specify the number of edges along which the paths  $p_{i(v)}$  and  $p_{j(v)}$  are being followed by some of the new paths.

Consider a node  $w$  in the tree  $T$  such that at least one edge in  $\bar{q}(w)$ , say an edge  $e$ , is saturated in the  $L$ -flow  $f$  (i.e.,  $f(e) = c(e)$ ). In this case, the properties of the tree  $T$  enforce that the node  $w$  has at least one 3-son  $u$  whose responsibility is to desaturate the edge  $e$  by diverting one of the paths that use  $e$  in  $f$  along a new route; the attribute  $j(u)$  specifies the index of the path from  $f$  that is being diverted by the 3-son  $u$  of  $w$  (Fig. 2), and  $h(w)$  specifies which saturated edges from  $\bar{q}(v)$  are desaturated by the son  $u$  of  $w$ .

As the definition of the tree  $T$  does not pose any requirements on the disjointness of the  $\bar{q}$ -paths corresponding to different nodes of  $T$ , it may happen that the paths  $\bar{q}(w)$  and  $\bar{q}(w')$  for two different nodes  $w$  and  $w'$  of the tree  $T$  overlap in a saturated edge  $e$ . In this case, Koubek and Říha allow an *exception* (our terminology) to the rule described in previous paragraph: if one of the nodes  $w$  and  $w'$ , say the node  $w$ , has a 3-son  $u$  that desaturates  $e$ , the other node, the node  $w'$ , need not have a 3-son but it may have a 4-son instead. The purpose of this 4-son

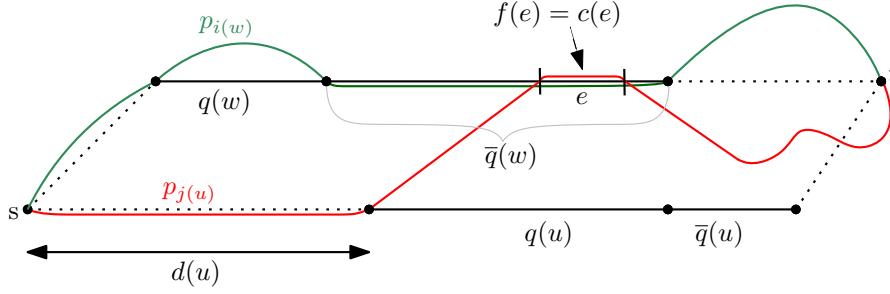


Figure 2: Desaturation of a saturated edge  $e$  in a  $\bar{q}(w)$  by a 3-son  $u$ .

is just to provide a pointer to the 3-son  $u$  of  $w$  that takes care about the desaturation of the edge  $e$ .

## 2.2 Small mistakes and typos

The paper is full of small mistakes and typos which change the meaning. Here we mention the most striking typo. On the page 393 in the paper [18], there is the rule 3b:

If  $v$  is a 1-son **of** a 3-son then  $v$  has a 1-son if and only if  
 $(END(q(v)) + b(v)) \bmod p_{i(v)} \neq t$ .

where  $v$  is node in the tree  $T$ ,  $t$  is a vertex in the graph  $G$ ,  $END(q(v))$  denotes the last vertex of the path  $q(v)$ , and for a path  $p$ , a vertex  $w$  on  $p$  and an integer  $k$ ,  $w + k \bmod p$  denotes the vertex on the path  $p$  that is  $k$  edges after  $w$ . The correct reading of the above rule, with a significantly different meaning, is:

If  $v$  is a 1-son **or** a 3-son, then  $v$  has a 1-son if and only if  
 $(END(q(v)) + b(v)) \bmod p_{i(v)} \neq t$ .

The difficulty with the original version is that it does not guarantee that the paths in new  $L$ -flow  $f'$  terminate in the vertex  $t$ .

## 2.3 The main error

We start by recalling a few definitions and lemmas from the original paper [18]; for the definition of the increasing system (more than one page long) we refer to [18].

**Definition 1** (Definition 4.2 in [18]). *Let  $T$  be an increasing  $L$ -system with respect to an  $L$ -flow  $f = \{(p_i, r_i) : i \in I\}$  in a network  $G = (X, R, c, s, t)$ . Given an edge  $u \in R$ , we define:*

- $T_1(u)$  is the number of vertices  $x$  in the tree  $T$  such that  $u \in \bar{q}(x)$  and if there is a saturated edge  $v \in \bar{q}(x)$  then there is a 3-son  $y$  of  $x$  with  $v \in h(y)$ ,  $u \notin p_{j(y)}$ .
- $T_2(u)$  is the number of vertices  $x$  in the tree  $T$  such that  $u \in q(x)$ .
- $T_3(u)$  is the number of vertices  $x$  which are 3-sons or 4-sons with  $u \in h(x)$ .

For  $i \in I$  we denote  $m_i = \sup\{T_3(u) : u \in p_i\}$ ,  $|T| = \min\{\frac{c(u)}{T_2(u)} : u \in R, f(u) = 0\} \cup \{\frac{c(u)-f(u)}{T_1(u)} : u \in R\} \cup \{\frac{r_i}{m_i} : i \in I\}$ , where the expressions that are not defined are omitted.

**Lemma 1** (Lemma 4.2 in [18]). *If there is an increasing  $L$ -system with respect to an  $L$ -flow  $f$ , then there is an  $L$ -flow  $g$  with  $|g| = |f| + |T|$ .*

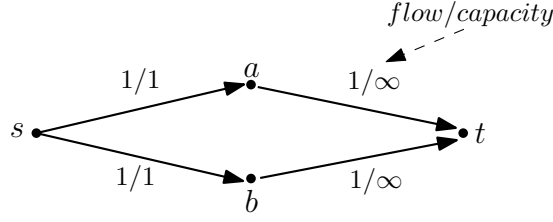


Figure 3: A network  $G$  with a 2-bounded flow  $f$ .

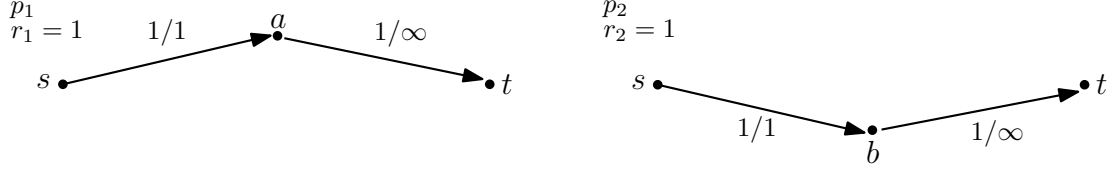


Figure 4: A decomposition of the 2-bounded flow  $f$  into paths  $p_1, p_2$ .

**Definition 2** (Definition 4.3 in [18]). Let  $\bar{R} = R \cup \{u'\}$ , where  $u' \notin R$  and  $c(u') = 0$ . We put  $\psi(G) = \min(|c(u) - c(v)| : c(u) \neq c(v), u, v \in \bar{R})$ .

**Lemma 2** (Lemma 4.4 in [18]). For each increasing  $L$ -system  $T$  (with respect to an  $L$ -flow  $f = \{(p_i, r_i) : i \in I\}$ ) constructed by the above procedure it holds  $|T| \geq \psi(G)/|I|$ .

The *above procedure* in Lemma 2 refers to a construction of an increasing  $L$ -system that is outlined in the original paper. As Definition 2 implies  $\psi(G) > 0$ , we also know by Lemma 2 that for every increasing  $L$ -system  $T$ ,  $|T| > 0$ .

Now we are ready to describe the counter example. Take  $k = 2$  and consider the following network  $G$  with a 2-bounded flow  $f$  of size 2 (Fig. 3 and 4); apparently, this is a maximum 2-bounded flow.

We are going to show that there exists an increasing system  $T$  for  $f$ . According to Lemmas 1 and 2 this implies the existence of a 2-bounded flow  $g$  of size  $|f| + |T| > |f|$ . As the flow  $f$  is a maximum 2-bounded flow in  $G$ , this is a contradiction.

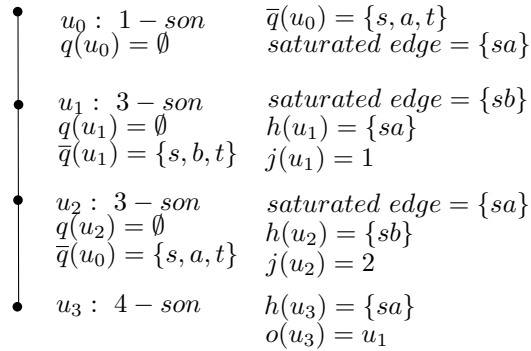


Figure 5: Increasing 2-system  $T$ .

The increasing system  $T$  is depicted in Figure 5; for the sake of simplicity, we list only the most relevant attributes. It is just a matter of a mechanical effort to check that it meets Definition 4.1 of the increasing system from the original paper.

In words, the essence of the counter example is the following. The purpose of the root of the tree, the node  $u_0$ , is to increase the flow from  $s$  to  $t$  along the path  $q(u_0)\bar{q}(u_0)$  which is

(accidentally) the path  $p_1$ . As there is a saturated edge on this path, namely the edge  $sa$ , there is a 3-son of the node  $u_0$ , the node  $u_1$ , whose purpose is to desaturate the edge  $sa$  by diverting one of the paths that use the edge  $sa$  along an alternative route; in particular, the node  $u_1$  is diverting the path  $p_1$  and it is diverting it from the very beginning, from  $s$ , along the path  $q(u_1)\bar{q}(u_1)$  which is (accidentally) the path  $p_2$ .

As there is a saturated edge on this path, namely the edge  $sb$ , there is a 3-son of the node  $u_1$ , the node  $u_2$ , whose purpose is to desaturate the edge  $sb$  by diverting one of the paths that use the edge  $sb$  along an alternative route; in particular, the node  $u_2$  is diverting the path  $p_2$  and it is diverting it from the very beginning, from  $s$ , along the path  $q(u_2)\bar{q}(u_2)$  which is (accidentally) again the path  $p_1$ .

As there is a saturated edge on this path, namely the edge  $sa$ , and as there is already another node in the tree that is desaturating  $sa$ , namely the node  $u_1$ , the node  $u_2$  does not have a 3-son but it has a 4-son  $u_3$  instead, which is a pointer to the 3-son  $u_1$ . This way, there is a kind of a deadlock cycle in the increasing system:  $u_1$  is desaturating the edge  $sa$  for the node  $u_0$  but it itself needs  $u_2$  to desaturate the edge  $sb$  in it and  $u_2$  in turn needs  $u_3$  to desaturate the edge  $sa$ , but  $u_3$  delegates this task back to  $u_1$ .

At this point, we know that Lemma 1 or Lemma 2 is not correct. By Definition 1, one can check that  $|T| = 1/2$  which implies, as we started with a maximum flow, that it is Lemma 1 that does not hold.

### 3 FPTAS for maximum $L$ -bounded flow

We first describe a fully polynomial approximation scheme for maximum  $L$ -bounded flow on networks with unit edge length. The algorithm is based on the algorithm for the maximum multicommodity flow by Garg and Könemann [11].

Then we describe a FPTAS for the  $L$ -bounded flow problem with general edge lengths. Our approximation schemas for the maximum  $L$ -bounded flow on unit edge lengths and the maximum  $L$ -bounded flow with edge lengths are almost identical, the only difference is in using an approximate subroutine for resource constrained shortest path in the general case which slightly complicates the analysis.

#### 3.1 FPTAS for Unit Edge Lengths

Let us consider the path based linear programming (LP) formulation of the maximum  $L$ -bounded flow,  $\mathbf{P}_{\text{path}}$ , and its dual,  $\mathbf{D}_{\text{path}}$ . We assume that  $G = (V, E, c, s, t)$  is a given network and  $L$  is a given length bound. Let  $\mathcal{P}_L$  denote the set of all  $s$ - $t$  paths of length at most  $L$  in  $G$ . There is a primal variable  $x(p)$  for each path  $p \in \mathcal{P}_L$ , and a dual variable  $y(e)$  for each edge  $e \in E$ . Note that the dual LP is a relaxation of an integer LP formulation of the minimum  $L$ -bounded cut problem.

$$\begin{array}{ll}
\max & \sum_{P \in \mathcal{P}_L} x(P) \\
\text{s.t.} & \sum_{\substack{P \in \mathcal{P}_L: \\ e \in P}} x(P) \leq c(e) \quad \forall e \in E \\
& x \geq 0
\end{array}
\qquad
\begin{array}{ll}
\min & \sum_{e \in E} c(e)y(e) \\
\text{s.t.} & \sum_{e \in P} y(e) \geq 1 \quad \forall P \in \mathcal{P}_L \\
& y \geq 0
\end{array}$$

The algorithm simultaneously constructs solutions for the maximum  $L$ -bounded flow and the minimum fractional  $L$ -bounded cut. It iteratively routes flow over shortest paths with respect to properly chosen dual edge lengths and at the same time increases these dual lengths; dual

edge length of the edge  $e$  after  $i$  iterations will be denoted by  $y_i(e)$ . During the runtime of the algorithm, the constructed flow need not respect the edge capacities; however, with the right choice of parameters  $\varepsilon, \delta$  the resulting flow can be scaled down to a feasible (i.e., respecting the edge capacities) flow (Lemma 3) that is a  $(1 + \varepsilon)$ -approximation of the maximum  $L$ -bounded flow (Theorem 1).

For a vector  $y$  of dual variables, let  $d_y^L(s, t)$  denote the length of the  $y$ -shortest  $s - t$  path from the set of paths  $\mathcal{P}_L$  and let  $\alpha^L(i) = d_{y_i}^L(s, t)$ . Note that a shortest  $s - t$  path with respect to edge lengths  $y$  that uses at most a given number of edges can be computed in polynomial time by a modification of the Dijkstra's shortest path algorithm.

---

**Algorithm 1** APPROX( $\varepsilon, \delta$ )

---

```

1:  $i \leftarrow 0, y_0(e) \leftarrow \delta \quad \forall e \in E, x_0(P) \leftarrow 0 \quad \forall P \in \mathcal{P}_L$ 
2: while  $\alpha^L(i) < 1$  do
3:    $i \leftarrow i + 1$ 
4:    $x_i \leftarrow x_{i-1}, y_i \leftarrow y_{i-1}$ 
5:    $P \leftarrow y_i$ -shortest  $s$ - $t$  path with at most  $L$  edges
6:    $c \leftarrow \min_{e \in P} c(e)$ 
7:    $x_i(P) \leftarrow x_i(P) + c$ 
8:    $y_i(e) \leftarrow y_i(e)(1 + \varepsilon c/c(e)) \quad \forall e \in P$ 
9: end while
10: return  $x_i$ 

```

---

Let  $f_i$  denote the size of the flow after  $i$  iterations,  $f_i = \sum_{P \in \mathcal{P}_L} x_i(P)$ , and let  $\tau$  denote the total number of iterations performed by APPROX; then  $x_\tau$  is the output of the algorithm and  $f_\tau$  its size.

**Lemma 3.** *The flow  $x_\tau$  scaled down by a factor of  $\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$  is a feasible  $L$ -bounded flow.*

*Proof.* By construction, for every  $i$ ,  $x_i$  is an  $L$ -bounded flow. Thus, we only have to care about the feasibility of the flow

$$\frac{x_\tau}{\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}}. \quad (1)$$

For every iteration  $i$  and every edge  $e \in E$ , as  $\alpha^L(i-1) < 1$ , we also have  $y_{i-1}(e) < 1$  and so  $y_i(e) < 1 + \varepsilon$ . It follows that

$$y_\tau(e) < 1 + \varepsilon. \quad (2)$$

Consider an arbitrary edge  $e \in E$  and suppose that the flow  $f_\tau(e)$  along  $e$  has been routed in iterations  $i_1, i_2, \dots, i_r$  and the amount of flow routed in iteration  $i_j$  is  $c_j$ . Then  $f_\tau(e) = \sum_{j=1}^r c_j$  and  $y_\tau(e) = \delta \prod_{j=1}^r (1 + \varepsilon c_j/c(e))$ . Because each  $c_j$  was chosen such that  $c_j \leq c(e)$ , we have by Bernoulli's inequality that  $1 + \varepsilon c_j/c(e) \geq (1 + \varepsilon)^{c_j/c(e)}$  and

$$y_\tau(e) \geq \delta \prod_{j=1}^r (1 + \varepsilon)^{c_j/c(e)} = \delta (1 + \varepsilon)^{f_\tau(e)/c(e)}. \quad (3)$$

Combining inequalities (2) and (3) gives

$$\frac{f_\tau(e)}{c(e)} \leq \log_{1+\varepsilon} \frac{1 + \varepsilon}{\delta}$$

which completes the proof. □



**Claim 4.** For  $i = 1, \dots, \tau$ ,

$$\alpha^L(i) \leq \delta L e^{\varepsilon f_i / \beta} . \quad (4)$$

*Proof.* For a vector  $y$  of dual variables, let  $D(y) = \sum_e c(e)y(e)$  and let  $\beta = \min_y D(y)/d_y^L(s, t)$ . Note that  $\beta$  is equal to the optimal value of the dual linear program. For notational simplicity we abbreviate  $D(y_i)$  as  $D(i)$ .

Let  $P_i$  be the path chosen in iteration  $i$  and  $c_i$  be the value of  $c$  in iteration  $i$ . For every  $i \geq 1$  we have

$$\begin{aligned} D(i) &= \sum_{e \in E} y_i(e)c(e) \\ &= \sum_{e \in E} y_{i-1}(e)c(e) + \varepsilon \sum_{e \in P_i} y_{i-1}(e)c_i \\ &= D(i-1) + \varepsilon(f_i - f_{i-1})\alpha^L(i-1) \end{aligned}$$

which implies that

$$D(i) = D(0) + \varepsilon \sum_{j=1}^i (f_j - f_{j-1})\alpha^L(j-1). \quad (5)$$

Now consider the length function  $y_i - y_0$ . Note that  $D(y_i - y_0) = D(i) - D(0)$  and  $d_{y_i - y_0}^L(s, t) \geq \alpha^L(i) - \delta L$ . Hence,

$$\beta \leq \frac{D(y_i - y_0)}{d_{y_i - y_0}^L(s, t)} \leq \frac{D(i) - D(0)}{\alpha^L(i) - \delta L} . \quad (6)$$

By combining relations (5) and (6) we get

$$\alpha^L(i) \leq \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^i (f_j - f_{j-1})\alpha^L(j-1) .$$

Now we define  $z(0) = \alpha^L(0)$  and for  $i = 1, \dots, \tau$ ,  $z(i) = \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^i (f_j - f_{j-1})z(j-1)$ . Note that for each  $i$ ,  $\alpha^L(i) \leq z(i)$ . Furthermore,

$$\begin{aligned} z(i) &= \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^i (f_j - f_{j-1})z(j-1) \\ &= \left( \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^{i-1} (f_j - f_{j-1})z(j-1) \right) + \frac{\varepsilon}{\beta} (f_i - f_{i-1})z(i-1) \\ &= z(i-1)(1 + \varepsilon(f_i - f_{i-1})/\beta) \\ &\leq z(i-1)e^{\varepsilon(f_i - f_{i-1})/\beta} . \end{aligned}$$

Since  $z(0) \leq \delta L$ , we have  $z(i) \leq \delta L e^{\varepsilon f_i / \beta}$ , and thus also, for  $i = 1, \dots, \tau$ ,  $\alpha^L(i) \leq \delta L e^{\varepsilon f_i / \beta}$ .  $\square$

**Theorem 1.** For every  $0 < \varepsilon < 1$  there is an algorithm that computes an  $(1 + \varepsilon)$ -approximation to the maximum  $L$ -bounded flow in a network with unit edge lengths in time  $\mathcal{O}(\varepsilon^{-2} m^2 L \log L)$ .

*Proof.* We start by showing that for every  $\varepsilon < \frac{1}{3}$  there is a constant  $\delta = \delta(\varepsilon)$  such that  $x_\tau$ , the output of  $\text{APPROX}(\varepsilon, \delta)$ , scaled down by  $\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}$  as in Lemma 3, is a  $(1 + 3\varepsilon)$ -approximation.

Let  $\gamma$  denote the approximation ratio of such an algorithm, that is, let  $\gamma$  denote the ratio of the optimal dual solution ( $\beta$ ) to the appropriately scaled output of  $\text{APPROX}(\varepsilon, \delta)$ ,

$$\gamma = \frac{\beta \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}}{f_\tau}, \quad (7)$$

where the constant  $\delta$  will be specified later.

By Claim 4 and the stopping condition of the while cycle we have

$$1 \leq \alpha^L(\tau) \leq \delta L e^{\varepsilon f_\tau / \beta}$$

and hence

$$\frac{\beta}{f_\tau} \leq \frac{\varepsilon}{\log \frac{1}{\delta L}}.$$

Plugging this bound in the equality for the approximation ratio  $\gamma$ , we obtain

$$\gamma \leq \frac{\varepsilon \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}}{\log \frac{1}{\delta L}} = \frac{\varepsilon}{\log(1+\varepsilon)} \frac{\log \frac{1+\varepsilon}{\delta}}{\log \frac{1}{\delta L}}.$$

Setting  $\delta = \frac{1+\varepsilon}{((1+\varepsilon)L)^{1/\varepsilon}}$  yields

$$\frac{\log \frac{1+\varepsilon}{\delta}}{\log \frac{1}{\delta L}} = \frac{\frac{1}{\varepsilon} \log((1+\varepsilon)L)}{(\frac{1}{\varepsilon} - 1) \log((1+\varepsilon)L)} = \frac{1}{1 - \varepsilon}.$$

Taylor expansion of  $\log(1 + \varepsilon)$  gives a bound  $\log(1 + \varepsilon) \geq \varepsilon - \frac{\varepsilon^2}{2}$  for  $\varepsilon < 1$  and it follows for  $\varepsilon < \frac{1}{3}$  that

$$\gamma \leq \frac{\varepsilon}{(1 - \varepsilon) \log(1 + \varepsilon)} \leq \frac{\varepsilon}{(1 - \varepsilon)(\varepsilon - \varepsilon^2/2)} \leq \frac{1}{1 - \frac{3}{2}\varepsilon} \leq 1 + 3\varepsilon.$$

To complete the proof, we just put  $\varepsilon' = \varepsilon/3$  and run  $\text{APPROX}(\varepsilon', \delta(\varepsilon'))$ . It remains to prove the time complexity of the algorithm. In every iteration  $i$  of  $\text{APPROX}$ , the length  $y_i(e)$  of an edge  $e$  with the smallest capacity on the chosen path  $P$  is increased by a factor of  $1 + \varepsilon'$ . Because  $P$  was chosen such that  $y_i(P) < 1$  also  $y_i(e) < 1$  for every edge  $e \in P$ . Lengths of other edges get increased by a factor of at most  $1 + \varepsilon'$ , therefore  $y_\tau(e) < 1 + \varepsilon'$  for every edge  $e \in E$ . Every edge has the minimum capacity on the chosen path in at most  $\left\lceil \log_{1+\varepsilon'} \frac{1+\varepsilon'}{\delta} \right\rceil = \mathcal{O}(\frac{1}{\varepsilon} \log_{1+\varepsilon} L)$  iterations, so  $\text{APPROX}$  makes at most  $\mathcal{O}(\frac{m}{\varepsilon} \log_{1+\varepsilon} L) = \mathcal{O}(\frac{m}{\varepsilon^2} \log L)$  iterations.

Each iteration takes time  $\mathcal{O}(Lm)$  so the total time taken by  $\text{APPROX}$  is  $\mathcal{O}(\varepsilon^{-2} m^2 L \log L)$ .  $\square$

### 3.2 FPTAS for General Edge Lengths

Now we extend the approximation algorithm to networks with general edge lengths that are given by a length function  $\ell : E \rightarrow \mathbb{N}$ . The dynamic programming algorithm for computing shortest paths that have a restricted length with respect to another length function, does not work in this case. In fact, the problem of finding shortest path with respect to a given edge length function while restricting to paths of bounded length with respect to another length function is NP-hard in general [13]. On the other hand, there exists a FPTAS for it [14, 20].

We assume that we are given as a black-box an algorithm that for a given graph  $G$ , two edge length functions  $y$  and  $\ell$ , two distinguished vertices  $s$  and  $t$  from  $G$ , a length bound  $L$  and an error parameter  $w > 0$ , computes a  $(1 + w)$ -approximation of the  $y$ -shortest path of

$\ell$ -length at most  $L$ ; we denote by  $d_{y,\ell}^L(s, t; w)$  the length of such a path and we also introduce an abbreviation  $\bar{\alpha}^L(i) = d_{y_i,\ell}^L(s, t; w)$ . Note that for every  $i$ ,  $\bar{\alpha}^L(i) \leq (1 + w)\alpha^L(i)$ . We can use the FPTAS of Lorenz and Raz [20] for this task.

The algorithm of Garg and Könemann [11] for approximating maximal multicommodity flow has been improved by Fleischer [9]. The original algorithm computes the shortest path between every terminal pairs in every iteration. Fleischer divided the algorithm to phases where she worked with commodities one by one. This way her algorithm effectively works with approximations of shortest paths while eliminates the dependency on the number of commodities and still gets a good approximation ratio. Using a similar analysis we show that we can work with an approximation shortest path algorithm to get an FPTAS to otherwise intractable maximum  $L$ -bounded flow problem with general edge lengths.

The structure of the  $L$ -bounded flow algorithm with general edge lengths stays the same as in the unit edge lengths case. The only difference is that instead of  $y$ -shortest  $L$ -bounded paths, approximations of  $y$ -shortest  $L$ -bounded paths are used (steps 2 and 5).

---

**Algorithm 2** APPROXGENERAL( $\varepsilon, \delta, w$ )

---

```

1:  $i \leftarrow 0, y_0(e) \leftarrow \delta \quad \forall e \in E, x_0(P) \leftarrow 0 \quad \forall P \in \mathcal{P}_L$ 
2: while  $\bar{\alpha}^L(i) < 1 + w$  do
3:    $i \leftarrow i + 1$ 
4:    $x_i \leftarrow x_{i-1}, y_i \leftarrow y_{i-1}$ 
5:    $P \leftarrow (1 + w)$ -approximation of the  $y_i$ -shortest  $L$ -bounded path
6:    $c \leftarrow \min_{e \in P} c(e)$ 
7:    $x_i(P) \leftarrow x_i(P) + c$ 
8:    $y_i(e) \leftarrow y_i(e)(1 + \varepsilon c/c(e)) \quad \forall e \in P$ 
9: end while
10: return  $x_i$ 

```

---

The analysis of the algorithm follows the same steps as the analysis of Algorithm 1 but one has to be more careful when dealing with the lengths.

As in the previous subsection, let  $f_i$  denote the size of the flow after  $i$  iterations and let  $\tau$  denote the total number of iterations. Due to the lack of space, the proofs are given in the Appendix.

**Lemma 5.** *The flow  $x_\tau$  scaled down by a factor of  $\log_{1+\varepsilon} \frac{(1+\varepsilon)(1+w)}{\delta}$  is a feasible  $L$ -bounded flow.*

*Proof.* For every edge  $e \in E$  and iteration  $i$ , as  $\bar{\alpha}^L(i-1) < 1 + w$ , we also have  $y_{i-1}(e) < 1 + w$ . By description of the algorithms, this implies  $y_i(e) < (1 + \varepsilon)(1 + w)$ , and in particular,

$$y_\tau(e) < (1 + \varepsilon)(1 + w) . \quad (8)$$

Combining this with  $y_\tau(e) \geq \delta(1 + \varepsilon)^{f_\tau(e)/c(e)}$  from inequality (3) in previous subsection, we derive

$$\frac{f_\tau(e)}{c(e)} \leq \log_{1+\varepsilon} \frac{(1 + \varepsilon)(1 + w)}{\delta}$$

which completes the proof. □

**Claim 6.** *For  $i = 1, \dots, \tau$ ,*

$$\alpha^L(i) \leq \delta L e^{\varepsilon(1+w)f_i/\beta} . \quad (9)$$

*Proof.* By the same reasoning as in the proof of Claim 4, we obtain

$$D(i) \leq D(0) + \varepsilon \sum_{j=1}^i (f_j - f_{j-1})(1+w)\alpha^L(i-1), \quad (10)$$

where the extra  $1+w$  factors stems from the fact that we work, in iteration  $i$ , not with a path of length  $\alpha(i)$  but with a path of length  $\bar{\alpha}(i) \leq (1+w)\alpha(i)$ . Combining this with  $\beta \leq \frac{D(i)-D(0)}{\alpha^L(i)-\delta L}$  from inequality (6), we obtain

$$\alpha^L(i) \leq \delta L + \frac{\varepsilon(1+w)}{\beta} \sum_{j=1}^i (f_j - f_{j-1})\alpha^L(j-1).$$

From this point, we proceed again along the same lines as in the proof of Claim 4 (the only difference is that instead of  $\varepsilon/\beta$ , we work now with  $(1+w)\varepsilon/\beta$ ) and get the desired bound.  $\square$

**Theorem 2.** *There is an algorithm that computes an  $(1+\varepsilon)$ -approximation to the maximum  $L$ -bounded flow in a graph with general edge lengths in time  $\mathcal{O}(\frac{m^2 n}{\varepsilon^2} \log L(\log \log n + \frac{1}{\varepsilon}))$ .*

*Proof.* We show that for every  $\varepsilon \leq \frac{1}{3}$  there are constants  $\delta$  and  $w$  such that  $x_\tau$ , the output of  $\text{APPROXGENERAL}(\varepsilon, \delta, w)$ , scaled down by  $\log_{1+\varepsilon} \frac{(1+\varepsilon)(1+w)}{\delta}$  as in Lemma 5, is a  $(1+5\varepsilon)$ -approximation to the maximum  $L$ -bounded flow with general capacities; the theorem easily follows.

Let  $\gamma$  denote the approximation ratio of such an algorithm, that is, let  $\gamma$  denote the ratio of the optimal dual solution ( $\beta$ ) to the appropriately scaled output of  $\text{APPROXGENERAL}(\varepsilon, \delta, w)$ ,

$$\gamma = \frac{\beta \log_{1+\varepsilon} \frac{(1+\varepsilon)(1+w)}{\delta}}{f_\tau}, \quad (11)$$

where the constants  $\delta$  and  $w$  will be specified later.

By the stopping condition of the while cycle we have  $1+w \leq \bar{\alpha}^L(\tau) \leq (1+w)\alpha^L(\tau)$ , that is,  $1 \leq \alpha^L(\tau)$ ; combining it with Claim 6, we get

$$\frac{\beta}{f_\tau} \leq \frac{\varepsilon(1+w)}{\log \frac{1}{\delta L}}.$$

Plugging this bound in the equality for the approximation ratio  $\gamma$ , we obtain

$$\gamma \leq \frac{\varepsilon(1+w) \log_{1+\varepsilon} \frac{(1+\varepsilon)(1+w)}{\delta}}{\log \frac{1}{\delta L}} = \frac{\varepsilon(1+w)}{\log(1+\varepsilon)} \frac{\log \frac{(1+\varepsilon)(1+w)}{\delta}}{\log \frac{1}{\delta L}}. \quad (12)$$

Setting  $\delta = \frac{(1+\varepsilon)(1+w)}{((1+\varepsilon)(1+w)L)^{1/\varepsilon}}$  yields

$$\frac{\log \frac{(1+\varepsilon)(1+w)}{\delta}}{\log \frac{1}{\delta L}} = \frac{\frac{1}{\varepsilon} \log((1+\varepsilon)(1+w)L)}{(\frac{1}{\varepsilon} - 1) \log((1+\varepsilon)(1+w)L)} = \frac{1}{1-\varepsilon}. \quad (13)$$

Thus, the bound on the approximation ratio  $\gamma$  (12) simplifies to

$$\gamma \leq \frac{\varepsilon(1+w)}{(1-\varepsilon) \log(1+\varepsilon)} \leq \frac{\varepsilon(1+w)}{(1-\varepsilon)(\varepsilon - \frac{\varepsilon^2}{2})} \leq \frac{1+w}{1 - \frac{3}{2}\varepsilon},$$

where the second inequality follows from the Taylor expansion of  $\log(1 + \varepsilon)$  and the bound  $\log(1 + \varepsilon) \geq \varepsilon - \frac{\varepsilon^2}{2}$ , for  $\varepsilon < 1$ . By setting  $w = \varepsilon$ , for  $\varepsilon \leq \frac{1}{3}$  we get the promised bound

$$\gamma \leq \frac{1 + w}{1 - \frac{3}{2}\varepsilon} \leq (1 + \varepsilon)(1 + 3\varepsilon) \leq 1 + 5\varepsilon .$$

Concerning the running time, we observe that in every iteration the length of at least one edge gets increased by the ratio  $1 + \varepsilon$ . For every edge  $e \in E$  we have  $y_\tau(e) \leq (1 + \varepsilon)(1 + w)$ . By the same arguments as in the previous subsection, our choice of the parameters ensures that the total number of iterations is at most  $\mathcal{O}(\frac{m}{\varepsilon} \log_{1+\varepsilon} L) = \mathcal{O}(\frac{m}{\varepsilon^2} \log L)$ . The FPTAS approximating the resource bounded shortest path takes time  $\mathcal{O}(mn(\log \log n + \frac{1}{\varepsilon}))$ . Combining these two bounds completes the proof.  $\square$

We note that the exponential length method can be used for many fractional packing problems and using the same technique we could get an approximation algorithm for maximum multicommodity  $L$ -bounded flow.

## References

- [1] J. Adánek and V. Koubek. Remarks on flows in network with short paths. *Comment. Math. Univ. Carolin.*, 12(4):661–667, 1971.
- [2] K. Altmanová. *Toky cestami omezené délky*. Bachelor’s thesis, Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, 2018. In Czech.
- [3] G. Baier. *Flows with path restrictions*. PhD thesis, TU Berlin, 2003.
- [4] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangráč, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010.
- [5] C. Bazgan, T. Fluschnik, A. Nichterlein, R. Niedermeier, and M. Stahlberg. A More Fine-Grained Complexity Analysis of Finding the Most Vital Edges for Undirected Shortest Paths. *CoRR*, abs/1804.09155, 2018.
- [6] A. Bley. On the complexity of vertex-disjoint length-restricted path problems. *Computational Complexity*, 12(3-4):131–149, 2003.
- [7] A. Bley and J. Neto. Approximability of 3- and 4-hop bounded disjoint paths problems. In F. Eisenbrand and F. B. Shepherd, editors, *Integer Programming and Combinatorial Optimization*, pages 205–218, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [8] P. Dvořák and D. Knop. Parametrized complexity of length-bounded cuts and multi-cuts. In *Theory and Applications of Models of Computation*, pages 441–452. Springer, 2015.
- [9] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discret. Math.*, 13(4):505–520, 2000.
- [10] T. Fluschnik, D. Hermelin, A. Nichterlein, and R. Niedermeier. Fractals for kernelization lower bounds, with an application to length-bounded cut problems. *CoRR*, abs/1512.00333, 2015.
- [11] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

- [12] P. A. Golovach and D. M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. In *International Symposium on Parameterized and Exact Computation*, 2009.
- [13] G. Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.
- [14] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.*, 17(1):36–42, 1992.
- [15] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982.
- [16] P. Kolman. On algorithms employing treewidth for  $L$ -bounded cut problems. *J. Graph Algorithms Appl.*, 22:177–191, 2018.
- [17] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. *J. Algorithms*, 61(1):20–44, 2006.
- [18] V. Koubek and A. Říha. The maximum  $k$ -flow in a network. In *Mathematical Foundations of Computer Science 1981*, pages 389–397. Springer, 1981.
- [19] E. Lee. Improved hardness for cut, interdiction, and firefighter problems. In *International Colloquium on Automata, Languages, and Programming*, 2017.
- [20] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Oper. Res. Lett.*, 28(5):213–219, 2001.
- [21] R. A. Mahjoub and T. S. McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Math. Program.*, 124(1-2):271–284, 2010.
- [22] B. Schieber, A. Bar-Noy, and S. Khuller. The Complexity of Finding Most Vital Arcs and Nodes. Technical report, College Park, MD, USA, 1995.
- [23] J. Voborník. *Algorithms for  $L$ -bounded flows*. Master’s thesis, Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, 2016.
- [24] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. The Computational Complexity of Finding Separators in Temporal Graphs. *ArXiv e-prints*, Nov. 2017.