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

May 28, 2020

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.

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. However, as far as we know, no polynomial-time combinatorial algorithm[1] for the L-bounded flow is known. For general edge lengths, the problem is NP-hard. 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 $\mathcal{O}(\varepsilon^{-2} m^2 L \log L)$ 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 Adámek 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 [8].

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

*This research was partially supported by project GA17-09142S of GA ČR.

[1]Combinatorial in the sense that it does not explicitly use linear programming methods or methods from linear algebra or convex geometry.

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 for the L-bounded flow is known.

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 $c : R \to \mathbb{R}^+$, 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 $G = (X, R \setminus C, c, s, t)$. In the *minimum L-bounded cut problem* the task 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 [5, 19, 5, 23]. 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 [20]. 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 [5] even in outer-planar graphs. Baier [4] 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 $L \geq 5$ [17], and the edge version is solvable in polynomial time for $L \leq 5$ and

NP-hard for $L \geq 6$ [7]. The polyhedra associated with L-bouded paths was studied by Dahl [9].

L-bounded cuts The L-bounded cut problem is NP-hard [24]. Baier et al. [5] 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 at al. [21] 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 [11, 26], too.

The best approximations that we are aware of are by Baier et al. [5]: 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 [5]. For the vertex version of the problem, there is a τ-approximation algorithm for graphs of treewidth τ [18].

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 [10, 18]. Golovach and Thilikos [14] 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 [18]. Fluschnik et al. [12] show that the L-bounded cut has no kernel of polynomial size when parameterized by L and the size of the cut (with reasonable complexity assumptions).

The L-bounded cut appears in the literature also as the short paths interdiction problem [6], [18], [21] or as the most vital edges for shortest paths [6].

1.2 Our contributions

In the first part of the paper, we show that the combinatorial algorithm by Koubek and Říha [20] 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 $\mathcal{O}(\varepsilon^{-2} m^2 L \log L)$; 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 [4].

Our result is not surprising (e.g., Baier [4] 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 [25]. A preliminary version of this work was presented at the 2019 WADS Algorithms and Data Structures Symposium [3].

2 The algorithm of Koubek and Říha

2.1 Increasing an L-bounded flow

The following notation is needed for the main definition of this subsection.

Definition 1 (Relevant parts of Definition 2.2 in [20]). *Given a directed graph (X, R), a* directed path of length n *from z_0 to z_n is a finite sequence $p = (z_0, u_1, z_1, \ldots, u_n, z_n)$, where $z_i \in X$ $(i = 0, 1, \ldots, n)$, $u_j \in R$ $(j = 1, 2, \ldots, n)$, $u_j = (z_{j-1}, z_J)$. We shall write $L(p) = n, BEG(p) = z_0, END(p) = z_n$. Whenever possible, we omit the edges in the notation of a path; we write e.g. $p = (z_0, z_1, \ldots, z_n)$.*
Given a directed path $p = (z_0, u_1, z_1, \ldots, u_n, z_n)$,

- *If $w = z_i$, then for any integer b, $-i \leq b \leq n - i$, we define $(x + b) \mod p = z_{i+b}$.*
- *If $i < j$, then $p|\{z_i, z_j\}$ is the directed path $(z_i, u_{i+1}, z_{i+1}, \ldots, u_j, z_j)$ of length $(j - i)$ from z_i to z_j.*

For the sake of completeness, we now proceed with the definition of an increasing L-system, a key notion in the paper by Koubek and Říha [20]. After the formal definition, we provide an informal explanation of the relevant parts of it. By $\mathbb{Z}^+$ we denote the set of all non-negative integers.

Definition 2 (Definition 4.1 in [20]). *Assume that f is an L-bounded flow from s to t in $G = (X, R, c, s, t)$ and $\{(p_i, r_i); i \in I\}$ a decomposition of f into paths of length at most L, path p_i carrying r_i units of flow, for each $i \in I$.* An increasing L-system with respect to the L-flow *f in the network G is a labeled oriented tree $T = (V, E, v_0, LABV, LABE)$, where*

a) V is the set of vertices, E is the set of edges, v_0 is the root of an oriented tree (V, E)

b) $LABV$ is a mapping labeling vertices: for each $v \in V$ $LABV(v) = ((q(v), i(v), a(v), b(v))$, where $q(v)$ is a path in G, $i(v) \in I$, $a(v), b(v) \in \mathbb{Z}^+$

c) $LABE$ is a partial mapping labeling edges: for each edge $u = (x, y)$ $LABE(u)$ is not defined or is equal to $(h(u), j(u), d(u), o(u))$, where $h(u) \in I$ or $h(u) \subset R$, $j(u) \in I$, $d(u) \in \mathbb{Z}^+$, $o(u) \in \mathbb{Z}^+$ or $o(u) \in V$ and if $h(u) \in I$ then $o(u) \in \mathbb{Z}^+$;
if $LABE(u)$ is undefined, then we say that y is a 1-son of x, if $h(u) \in I$, $o(u) \in \mathbb{Z}^+$ then y is a 2-son of x, if $h(u) \subset R$, $o(u) \in \mathbb{Z}^+$ then y is a 3-son of x, if $h(u) \subset R$, $o(u) \in V$ then y is a 4-son of x; as (V, E) is a tree we get that an edge $u = (x, y)$ is uniquely determined by its end-vertex y, and therefore we shall often write for $y \in V$ $LABE(y) = (h(y), j(y), d(y), o(y))$ instead of $LABE(u)$ where $u = (x, y)$;

and for values of $LABV$, $LABE$ the following conditions hold:

1) for each $v \in V$:

- *if $u \in q(v)$ then $f(u) = 0$,*
- *$END(q(v)) \in p_{i(v)}$,*
- *$a(v) + L(q(v)) + b(v) \leq L$ and $a(v) + L(q(v)) + b(v) = L$ implies $(END(q(v)) + b(v)) \mod p_{i(v)} = t$;*

2) $BEG(q(v_0)) = s$, $a(v_0) = 0$ and either $L(q(v_0)) > 0$ or $u = (s, (s + 1) \mod p_{i(v_0)})$ is not saturated;

3) for each $v \in V$:

a) if v is a 4-son then v is a leaf of the tree (V, E)

b) if v is a 1-son or a 3-son then v has a 1-son iff $(END(q(v)) + b(v)) \mod p_{i(v)} \neq t$

c) v has at most one 1-son and at most one 2-son

d) v has a 2-son iff v is a 2-son or a 3-son and $d(v) > o(v) > 0$

e) v has a 3-son or a 4-son iff there is $u \in \bar{q}(v)$ with $f(u) = c(u)$, and where here and in what follows $\bar{q}(v) = p_{i(v)} \mid \{END(q(v)), (END(q(v)) + b(v)) \mod p_{i(v)}\}$;

4) if v is a 1-son of w then $BEG(q(v)) = (END(q(w) + b(w)) \mod p_{i(w)}$, $a(v) = a(w) + L(q(w)) + b(w)$;

5) if v is a 2-son of w then $d(w) - o(w) \geq d(v) - o(v)$, $a(v) = o(v)$, $a(v) = L(q(v)) + b(v) = a(w)$, $(s + d(v)) \mod p_{j(v)} = BEG(q(v))$, $i(v) = h(v)$;

6) if v is a 3-son of w then $h(v) \subseteq \{u;\ u \in \bar{q}(w), f(u) = c(u)\} \cap p_{j(v)}$, $h(v) \neq \emptyset$, $a(v) = o(v)$, $BEG(q(v)) = (s + d(v)) \mod p_{j(v)}$ precedes every edge $u \in h(v)$;

7) if v is a 4-son of w then $j(v) = j(o(v))$, $o(v)$ is a 3-son, $\emptyset \neq h(v) \subseteq \{u;\ u \in \bar{q}(w),\ f(u) = c(u)\} \cap p_{j(v)}$, and the following condition hold: let $u_1, \ldots, u_n$ $(v_1, \ldots, v_m)$ be the sequence of vertices of the tree (V, E) such that for $i = 1, \ldots, n-1$ $(j = 1, \ldots, m-1)$ u_{i+1} is a 2-son of u_i (v_{j+1} is a 1-son of v_j), $u_1 = v_1 = o(v)$, u_n (v_m) has no 2-son (1-son); then $z \notin p_{j(u_m)} | \{s, s + o(u_n\}$ and $z \notin \bar{q}(u_i)$, $z \notin \bar{q}(v_j)$ for each $z \in h(v)$, $1 < i \leq n$, $1 < j \leq m$;

8) for each vertex v: if $Y(v)$ is the set of all 3-sons and 4-sons of v then $\{u;\ u \in \bar{q}(v),\ f(u) = c(u)\} = \bigcup h(w)$ where the union is taken over all $w \in Y(v)$; if $w_1 \neq w_2$, $w_1, w_2 \in Y(v)$, then $h(w_1) \cap h(w_2) = \emptyset$;

9) for every path p in (V, E) and for every couple of vertices $v_1, v_2 \in p$, $v_1 \neq v_2$, v_1, v_2 being a 3-son or a 4-son, it holds $h(v_1) \cap h(v_2) = \emptyset$.

The algorithm of Koubek and Řríha [20] is supposed to work as follows: given an arbitrary L-flow f from s to t in G that is not a maximum L-flow, build an increasing L-system $T = (V, E, v_0, LABV, LABE)$ and use it to derive a larger L-flow f' from the L-flow f (cf. Lemma 1). In the rest of this subsection we provide an informal description of the meaning of various attributes of the increasing L-system.

For (almost) each node u in T, there are two consecutive paths in G associated with: 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 of these paths with paths in f and this combination

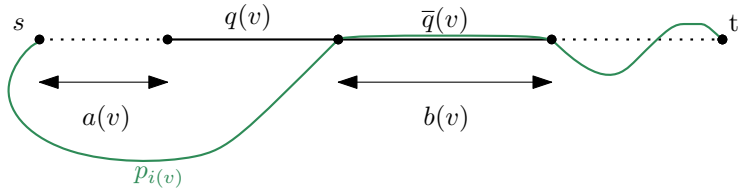

Figure 1: The paths $q(v)$ and $\bar{q}(v)$ associated with the node v.

is supposed to yield the larger L-flow f'. To explain the error in the paper, it is sufficient to deal only with three of the four types of nodes in T, namely with types 1-son, 3-son and 4-son.

The attributes $a(v)$ and $d(v)$ of the node labels 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 $b(v)$ specifies the number of edges along which the path $p_{i(v)}$ is being followed by $\bar{q}(v)$.

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 node u (Fig. 2), and $h(u)$ specifies which

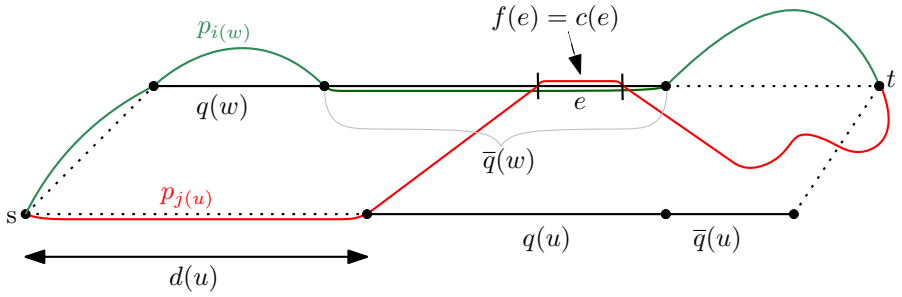

Figure 2: Desaturation of a saturated edge e in a $\bar{q}(w)$ by a 3-son u.

saturated edge(s) from $\bar{q}(w)$ are desaturated this way by the node u.

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 such a case, Koubek and Ř́iha 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, and if w' is not a descendant of u, then the node w' need not have a 3-son for desaturation of e but it may have a 4-son instead. The purpose of this 4-son 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 The main error

We start by recalling a few more definitions and lemmas from the original paper [20]. In this section we identify an L-bounded flow f with its decomposition $\{(p_i, r_i); i \in I\}$.

Definition 3 (Definition 4.2 in [20]). *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 $u \in \overline{q}(x)$ and if there is a saturated edge $v \in \overline{q}(x)$ 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 [20]). *If there is an increasing L-system with respect to an L-flow f, then there is an L-flow g with $|g| = |f| + |T|$.*

Definition 4 (Definition 4.3 in [20]). *Let $\overline{R} = R \cup \{u'\}$, where $u' \notin R$ and $c(u') = 0$. We put $\psi(G) = \min(|c(u) - c(v)| : c(u) \neq c(v), u, v \in \overline{R})$.*

Lemma 2 (Lemma 4.4 in [20]). *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 $|T| \geq \psi(G)/|I|$.*

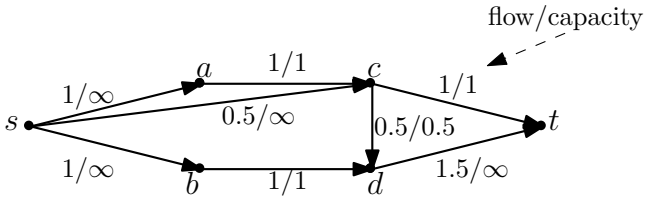

Figure 3: A network G with a 4-bounded flow f.

The *above procedure* in Lemma 2 refers to a construction of an increasing L-system that is outlined in the original paper. As Definition 4 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.

Lemma 3. *There exist a network G, a maximum L-flow f in G and an increasing L-system T with respect to f.*

Proof. Take $L = 4$ and let G be a network $G = (X, R, c, s, t)$ defined as follows: $X = \{s, t, a, b, c, d\}$, $R = \{(s, a), (s, b), (s, c), (a, c), (b, d), (c, d), (c, t), (d, t)\}$, $c(a, c) = c(b, d) = c(c, t) = 1$, $c(c, d) = 1/2$ and all other edges have unbounded capacity. Consider a 4-flow f defined by the following path decomposition: $p_0 = (s, c, t)$, $p_1 = (s, a, c, t)$, $p_2 = (s, b, d, t)$, $p_3 = (s, a, c, d, t)$ and $r_0 = r_1 = r_3 = 1/2$ and $r_2 = 1$; note that f is a maximum 4-flow between s and t.

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 4-bounded flow g of size $|f| + |T| > |f|$. As the flow f is a maximum 4-bounded flow in G, this is a contradiction.

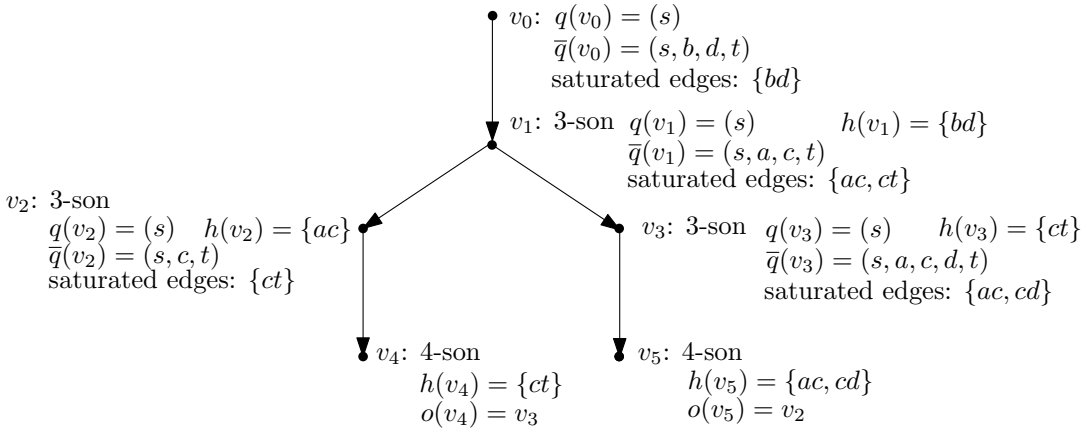

Figure 4: Increasing 4-system T. *Saturated edges* are the edges from $\overline{q}$ that are saturated in f.

The increasing system T is depicted in Figure 4 and described in detail in Table 1. It is just a matter of a mechanical effort to check that it meets Definition 2[2] □

In words, the essence of the counter example is the following. The purpose of the root of the tree, the node v_0, is to increase the flow from s to t along the path $q(v_0)\overline{q}(v_0) = (s, b, d, t)$. As there is an edge saturated in f on this path, namely the edge bd, there is a 3-son of the node v_0, the node v_1, whose purpose is to desaturate the edge bd by diverting one of the paths that

[2]Due to an attempt for simpicity, the counter-example given in the preliminary version of the paper [3] is erroneous - it does not satisfy the property 9.

	q	$\bar{q}$	i	a	b	h	j	d	o	type
v_0	(s)	(s,b,d,t)	2	0	3	–	–	–	–	1-son
v_1	(s)	(s,a,c,t)	1	0	3	$\{bd\}$	2	0	0	3-son
v_2	(s)	(s,c,t)	0	0	2	$\{ac\}$	3	0	0	3-son
v_3	(s)	(s,a,c,d,t)	3	0	4	$\{ct\}$	1	0	0	3-son
v_4	(s)	(s)	1	0	0	$\{ct\}$	1	0	v_3	4-son
v_5	(s)	(s)	1	0	0	$\{ac,cd\}$	3	0	v_2	4-son

Table 1: The labels of the increasing 4-system T.

use it in f along an alternative route; in particular, the node v_1 is diverting the path $p_{j(v_1)} = p_2$ and it is diverting it from the very beginning, from s, along the path $q(v_1)\bar{q}(v_1) = (s,a,c,t)$.

As there are two edges saturated in f on this path, namely the edges ac and ct, there are two 3-sons v_2 and v_3 of the node v_1. The purpose the node v_2 is to desaturate the edge ac by diverting one of the paths that use it along an alternative route and, similarly, the purpose the node v_3 is to desaturate the edge ct by diverting one of the paths that use it along an alternative route. In particular, the node v_2 is diverting the path $p_{j(v_2)} = p_3$ and it is diverting it along the path $q(v_2)\bar{q}(v_2) = (s,c,t)$, and the node v_3 is diverting the path $p_{j(v_3)} = p_1$ along the path $q(v_3)\bar{q}(v_3) = (s,a,c,d,t)$.

As there is a saturated edge on the path (s,c,t), namely the edge ct, and as there is already another node in the tree that is desaturating ct, namely the node v_3, the node v_2 does not have a 3-son but it has a 4-son v_4 instead, which is just a pointer to the node v_3. Similarly, as there is a saturated edge on the path (s,a,c,d,t), namely the edge ac, and as there is already another node in the tree that is desaturating ac, namely the node v_2, the node v_3 does not have a 3-son but it has a 4-son v_5 instead, which is just a pointer to the node v_3; the diversion of the path $p_{j(v_5)} = p_3$ will desaturate also the edge cd.

This way, there is a kind of a deadlock cycle in the increasing system: the task of v_4 is to desaturate the edge ct for the node v_2 but it itself needs v_3 to do it; v_3 in turn needs v_5 to desaturate the edge ac, but v_5 delegates this task back to v_2. Thus, none of the nodes does the real desaturation that is needed for the increase of the flow.

Corollary 4. *The algorithm for maximum L-bounded flow [20] does not work.*

At this point, we know that Lemma 1 or Lemma 2 is not correct. By Definition 3, one can check that $|T| > 0$ 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 primal-dual algorithm for the maximum multicommodity flow by Garg and Könemann [13].

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.

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

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 $y_i(e)$. The progress of the algorithm depends on two positive parameters, $\varepsilon < 1$, $\delta < 1$. During the runtime of the algorithm, the constructed flow need not respect the edge capacities; however, with the right choice of parameters ε, δ the resulting flow can be scaled down to a feasible (i.e., respecting the edge capacities) flow (Lemma 5) that is a $(1+\varepsilon)$-approximation of the maximum L-bounded flow (Theorem 7).

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(ε, δ)

$i \leftarrow 0,\ y_0(e) \leftarrow \delta \quad \forall e \in E,\ x_0(P) \leftarrow 0 \quad \forall P \in \mathcal{P}_L$
while $\alpha^L(i) < 1$ **do**
  $i \leftarrow i + 1$
  $x_i \leftarrow x_{i-1}, y_i \leftarrow y_{i-1}$
  $P \leftarrow y_i$-shortest s-t path with at most L edges
  $c \leftarrow \min_{e \in P} c(e)$
  $x_i(P) \leftarrow x_i(P) + c$
  $y_i(e) \leftarrow y_i(e)(1 + \varepsilon c / c(e)) \quad \forall e \in P$
end while
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 τ denote the total number of iterations performed by APPROX; then x_τ is the output of the algorithm and f_τ its size.

Lemma 5. *The flow x_τ 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

$$\frac{x_\tau}{\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}} \ . \tag{1}$$

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

$$y_\tau(e) < 1 + \varepsilon \ . \tag{2}$$

Consider an arbitrary edge $e \in E$ and suppose that the flow $f_\tau(e)$ along e has been routed in iterations $i_1, i_2, \ldots, i_r$ 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 $c_j \le c(e)$, we have by Bernoulli's inequality that $1 + \varepsilon c_j/c(e) \ge (1+\varepsilon)^{c_j/c(e)}$ and

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

Combining inequalities (2) and (3) gives

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

which completes the proof. □

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

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

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 β 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 $i \ge 1$ 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

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

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

$$\beta \le \frac{D(y_i - y_0)}{d^L_{y_i - y_0}(s,t)} \le \frac{D(i) - D(0)}{\alpha^L(i) - \delta L} \ . \tag{6}$$

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

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

Now we define $z(0) = \alpha^L(0)$ and for $i = 1, \ldots, \tau$, $z(i) = \delta L + \frac{\varepsilon}{\beta} \sum_{j=1}^{i} (f_j - f_{j-1}) z(j-1)$. 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 $i = 1, \ldots, \tau$, $\alpha^L(i) \leq \delta L e^{\varepsilon f_i/\beta}$. □

Theorem 7. *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 $\delta = \delta(\epsilon)$ such that x_τ, the output of Approx(ε, δ), scaled down by $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ as in Lemma 5, is a $(1+3\varepsilon)$-approximation.

Let γ denote the approximation ratio of such an algorithm, that is, let γ denote the ratio of the optimal dual solution (β) to the appropriately scaled output of Approx(ε, δ),

$$\gamma = \frac{\beta \log_{1+\epsilon} \frac{1+\epsilon}{\delta}}{f_\tau} , \tag{7}$$

where the constant δ will be specified later.

By Claim 6 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 γ, 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

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

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 Approx$(\varepsilon', \delta(\varepsilon'))$. It remains to prove the time complexity of the algorithm. In every iteration i of 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 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 Approx is $\mathcal{O}(\varepsilon^{-2} m^2 L \log L)$. □

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 $\ell : E \to \mathbb{N}$. 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 [15]. On the other hand, there exists a FPTAS for it [16, 22].

We assume that we are given as a black-box an algorithm that for a given graph G, two edge length functions y and ℓ, 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 ℓ-length at most L; we denote by $d^L_{y,\ell}(s, t; w)$ the length of such a path and we also introduce an abbreviation $\bar{\alpha}^L(i) = d^L_{y_i,\ell}(s, t; w)$. Note that for every i, $\bar{\alpha}^L(i) \leq (1 + w)\alpha^L(i)$. We can use the FPTAS of Lorenz and Raz [22] for this task.

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(ε, δ, 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: $\quad i \leftarrow i + 1$
4: $\quad x_i \leftarrow x_{i-1}, y_i \leftarrow y_{i-1}$
5: $\quad P \leftarrow (1 + w)$-approximation of the y_i-shortest L-bounded path
6: $\quad c \leftarrow \min_{e \in P} c(e)$
7: $\quad x_i(P) \leftarrow x_i(P) + c$
8: $\quad y_i(e) \leftarrow y_i(e)(1 + \varepsilon c / c(e)) \quad \forall e \in P$
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 τ denote the total number of iterations performed by ApproxGeneral; then x_τ is the output flow and f_τ its size.

Lemma 8. *The flow x_τ 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,

$$y_\tau(e) < (1 + \varepsilon)(1 + w) \ . \tag{8}$$

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 9. *For $i = 1, \ldots, \tau$,*

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

Proof. By the same reasoning as in the proof of Claim 6, we obtain

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

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 6 (the only difference is that instead of ϵ/β, we work now with $(1+w)\epsilon/\beta$) and get the desired bound. □

Theorem 10. *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 δ and w such that x_τ, the output of ApproxGeneral(ε, δ, w), scaled down by $\log_{1+\varepsilon} \frac{(1+\varepsilon)(1+w)}{\delta}$ as in Lemma 8, is a $(1+5\varepsilon)$-approximation to the maximum L-bounded flow with general capacities; the theorem easily follows.

Let γ denote the approximation ratio of such an algorithm, that is, let γ denote the ratio of the optimal dual solution (β) to the appropriately scaled output of ApproxGeneral(ε, δ, w),

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

where the constants δ 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 9, 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 γ, we obtain

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

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

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

Thus, the bound on the approximation ratio γ (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. □

4 Conclusion and Open Problems

The maximum L-bounded flow problem looks as a simple modification of the maximum flow problem. We know that it is solvable in polynomial time using LP algorithms. However, it is not obvious how to solve it by combinatorial algorithms (though, e.g., the Dinic's algorithm for maximum flow implicitly deals with lengths of flow paths) and currently, no such algorithm is known, despite the effort to find some. The best we can do without LP algorithms is the FPTAS described in this paper.

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.

It is a challenging open problem to design an exact polynomial time combinatorial algorithm for the maximum L-bounded flow. Considering the fact that one of the first algorithms for the maximum flow problem was a primal-dual algorithm, a more specific question is whether we can solve the maximum L-bounded problem exactly by a primal-dual algorithm.

References

[1] J. Adámek 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. Technical report, Bachelor's thesis, Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, in Czech, 2018.

[3] K. Altmanová, P. Kolman, and J. Voborník. On polynomial-time combinatorial algorithms for maximum l-bounded flow. In *Algorithms and Data Structures*, pages 14–27. Springer, 2019.

[4] G. Baier. *Flows with path restrictions*. PhD thesis, TU Berlin, 2003.

[5] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010.

[6] 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.

[7] A. Bley. On the complexity of vertex-disjoint length-restricted path problems. *Computational Complexity*, 12(3-4):131–149, 2003.

[8] 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.

[9] G. Dahl. Notes on polyhedra associated with hop-constrained paths. *Operations Research Letters*, 25(2):97–100, 1999.

[10] 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.

[11] 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.

[12] T. Fluschnik, D. Hermelin, A. Nichterlein, and R. Niedermeier. Fractals for kernelization lower bounds. *SIAM J. Discrete Math*, 32(1):656–681, 2018.

[13] 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.

[14] 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.

[15] G. Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.

[16] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.*, 17(1):36–42, 1992.

[17] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982.

[18] P. Kolman. On algorithms employing treewidth for L-bounded cut problems. *J. Graph Algorithms Appl.*, 22:177–191, 2018.

[19] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. *J. Algorithms*, 61(1):20–44, 2006.

[20] V. Koubek and A. Říha. The maximum k-flow in a network. In *Mathematical Foundations of Computer Science 1981*, pages 389–397. Springer, 1981.

[21] E. Lee. Improved hardness for cut, interdiction, and firefighter problems. In *International Colloquium on Automata, Languages, and Programming*, 2017.

[22] 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.

[23] 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.

[24] B. Schieber, A. Bar-Noy, and S. Khuller. The Complexity of Finding Most Vital Arcs and Nodes. Technical report, College Park, MD, USA, 1995.

[25] J. Voborník. Algorithms for L-bounded flows. Master's thesis, Charles University, Faculty of Mathematics and Physics, Department of Applied Mathematics, 2016.

[26] P. Zschoche, T. Fluschnik, H. Molter, and R. Niedermeier. The Computational Complexity of Finding Separators in Temporal Graphs. *ArXiv e-prints*, Nov. 2017.