Non-approximability and Polylogarithmic Approximations of the Single-Sink Unsplittable and Confluent Dynamic Flow Problems*†

Mordecai J. Golin[1], Hadi Khodabande[2], and Bo Qin[3]

1 **CSE Department, The Hong Kong University of Science and Technology, Hong Kong**
golin@cse.ust.hk

2 **CE Department, Sharif University of Technology, Teheran, Iran**
khodabande@ce.sharif.edu

1 **CSE Department, The Hong Kong University of Science and Technology, Hong Kong**
bqin@cse.ust.hk

Abstract

Dynamic Flows were introduced by Ford and Fulkerson in 1958 to model flows over time. They define edge *capacities* to be the total amount of flow that can enter an edge *in one time unit*. Each edge also has a *length*, representing the time needed to traverse it. Dynamic Flows have been used to model many problems including traffic congestion, hop-routing of packets and evacuation protocols in buildings. While the basic problem of moving the maximal amount of supplies from sources to sinks is polynomial time solvable, natural minor modifications can make it NP-hard. One such modification is that flows be *confluent,* i.e., all flows leaving a vertex must leave along the same edge. This corresponds to natural conditions in, e.g., evacuation planning and hop routing.

We investigate the *single-sink Confluent Quickest Flow* problem. The input is a graph with edge capacities and lengths, sources with supplies and a sink. The problem is to find a confluent flow minimizing the time required to send supplies to the sink. Our main results include:

- *Logarithmic Non-Approximability.* Directed Confluent Quickest Flows cannot be approximated in polynomial time with an $O(\log n)$ approximation factor, unless $P = NP$.
- *Polylogarithmic Bicriteria Approximations.* Polynomial time $(O(\log^8 n), O(\log^2 \kappa))$ bicriteria approximation algorithms for the Confluent Quickest Flow problem where κ is the number of sinks, in both directed and undirected graphs.

Corresponding results are also developed for the *Confluent Maximum Flow over time* problem. The techniques developed also improve recent approximation algorithms for *static* confluent flows.

1998 ACM Subject Classification G.1.6, Optimization, G.2.1, Combinatorics, G.2.2 Graph Theory

Keywords and phrases Optimization, Approximation, Dynamic Flow, Confluent Flow

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.41

* The work of all three authors was partially supported by RGC Hong Kong CERG grant 16208415.
† A full version of the paper is available at [7], https://arxiv.org/abs/1709.10307.

1 Introduction

Note: Due to space considerations, this extended abstract is missing many of the proofs of the theorems and lemmas stated. For complete proofs and accompanying diagrams, please see the full version of this paper at [7].

Network Flow problems are very well known. Their input is a graph network with *capacities* $c(e)$ on its edges. $c(e)$ is the maximum *flow* that can be pushed through e. The problem is usually to maximize the amount of flow that can be pushed through the network. By contrast, *Dynamic* network flows, introduced by Ford and Fulkerson [5] in 1958, around the same time as regular network flows, are not as well known. In Dynamic Flows, $c(e)$ becomes the amount of flow *that can enter* e *in one time unit* while edge *length* $\ell(e)$ is the time that it takes for a unit of flow to traverse e. Dynamic Flow problems need to consider the additional problem of *congestion*, which may arise while flow waits to enter an edge.

Dynamic flows have been used to model problems as diverse as traffic movement, evacuation protocols and hop-routing of packets. The *(Dynamic) Maximum Flow Over Time* problem is to find the maximum amount of flow that can be pushed from sources to sinks in a given amount of time. The *(Dynamic) Quickest Flow* problem is to find the minimum time in which a fixed amount of flow can be pushed from sources to sinks. In addition, there are *multicommodity-flow* versions which require specific amounts of flow between given source-sink pairs and *transshipment problems* versions which do not restrict which source's demands are pushed to which sinks. It is known that the *Quickest Multicommodity Flow Over Time* problem is NP-Hard [9] while the *Quickest Transshipment* problem can be solved in polynomial time [11, 12]. Good surveys on Dynamic Flow problems and an introduction to its basic literature can be found in [15, 19, 23].

In basic (static) network flow problems, *splittable* flow is permitted, i.e., flow between a source and sink can be divided into multiple parts with each being routed over a different path. *Unsplittable flows* require that all flow between a particular source and sink be routed over only one path. *Confluent flows* require that all flow passing through a vertex must leave that vertex on the same edge[1] [2, 22]. Very recent work [21] has shown that, for the static single-sink case, unless $P = NP$, optimal unsplittable flows and optimal confluent flows do not have polynomial time constant-factor approximation[2] algorithms and, in fact, confluent flows can not be approximated to within a factor of $O(m^{1/2-\epsilon})$.

Confluent flows were introduced by [3], with applications including Internet routing [1], evacuation problems [17], and traffic coordination [15]. Several works have studied confluent flows that minimize the maximum *congestion* in routing networks e.g., [3, 2, 22]. However, these works usually do not take into consideration the transit time (or edge length) required for a packet to traverse a single link, though this parameter is usually considered in general network analyses (see, e.g., [10]). This immediately raises the Confluent Quickest Flow problem: Does there exists any routing scheme that minimizes the total time for sending all packets via a feasible (congestion bounded) confluent flow?

Another scenario in which *confluent dynamic flows* arise naturally is in modelling evacuation protocols. Let vertices represent locations to be evacuated and edges represent paths between vertices. A vertex's original supply is the number of people to be evacuated from it and a sink corresponds to an emergency exit. $\ell(e)$ is the time required to traverse path e; $c(e)$ is the number of people that can enter e in parallel, i.e., its width. The Confluent Flow

[1] Thus, confluent flows partition flows into edge disjoint in-trees, with the root of each tree being a sink.
[2] The objectives studied in [21] are the total *amount* of flow that can be confluently routed or the number of demands that can be confluently satisfied in the *static* flow.

restriction states that all people passing through a vertex must leave by the same edge, i.e., following a sign pointing "This way out". The Quickest Flow problem corresponds to placing the exit signs so as to minimize the time required to evacuate all people. The Maximum Flow Over Time problem corresponds to placing the signs so as to maximize the number of people that can be evacuated in a given amount of time.

The single-source single-sink version of the Confluent Quickest Flow problem is the polynomial-time solvable [19] *Quickest-Path Problem*. The Confluent Flow version of the multiple-source multiple-sink Quickest Transshipment problem was known to be polynomial-time solvable when G is a tree [17]. It was also known that, for general graphs, the single-sink Confluent Quickest Transshipment problem is NP-Hard [13]. But no other hardness complexity results, and in particular, non-approximability results, were known for general G.

Our first results are that *Confluent Dynamic Flow* problems on directed graphs, both the Quickest Flow and Max Flow Over Time versions, cannot be approximated to within $O(\log n)$ (n being the number of vertices in G) unless $P = NP$. Our results hold even when the graph has *a single sink*. Since, Multicommodity Flow and Transshipment are equivalent in the single-sink case we write "Quickest Flow" instead of "Quickest Multicommodity Flow" or "Quickest Transshipment".

In the other direction, we present *polylogarithmic bicriteria approximation* algorithms for both the *single-sink* Confluent Quickest Flow and Confluent Maximum Flow Over Time problems, in both directed and undirected networks. Note that known approximation algorithms for confluent flows are restricted to static networks in [3, 2, 22], and known optimal algorithms for dynamic confluent flows are restricted to special graphs, e.g., trees [17]. To the best of our knowledge, our algorithm is the first polylogarithmic approximation for these problems in general networks. These results are presented in Tables 1-2.

1.1 Single-Sink Dynamic Unsplittable/Confluent Flow Problems

The input to the problems is a dynamic flow network, i.e., a graph $G = (V, E)$ with n nodes and m edges, where edge e has capacity $c(e)$ and length $\ell(e)$. Also specified are a collection of sources $\{s_1, ..., s_k\} \subset V$ and a sink $t \in V$. The problems studied are:

- Quickest Flow Problem: Provides additional inputs $\{d_1, ..., d_\kappa\}$. d_i is the supply at source s_i. The problem is to find a flow minimizing the time it takes to send all of the d_i units of supply to sink t.
- Maximum Flow Over Time Problem: Provides additional input of time horizon T. The problem is to find a flow maximizing the amount of supply sent to the sink t within time horizon T. Supply at the s_i is unlimited.

We treat two different types of flow restrictions:

- *Unsplittable Flow:* All flow from s_i to t must pass along the same path P_i from s_i to t.
- *Confluent Flow:* Any two supplies that meet at a node must traverse an identical path to the sink t. In particular, at most one edge out of each node v is allowed to carry flow. Consequently, the support of the flow is a tree with all paths in the tree terminating at t.

1.2 Our Results

Section 3.1 presents a simple proof that, unless $P = NP$, $\forall \epsilon > 0$, it is impossible to construct a polynomial-time $3/2 - \epsilon$ approximation algorithm for the single-sink Quickest Flow problem when flows are restricted to be either unsplittable or confluent. This result holds for both directed and undirected graphs and even when the graph is restricted to have only one sink.

Table 1 Hardness or lower bounds on approx. ratio for the single-sink Quickest Flow problem.

Flow	Dynamic Network	Hardness or LB on Approx. Ratio
Confluent	Trees	Polynomial-Time Solvable [17]
Confluent	Directed/Undirected	NP-Hard [13]
Unsplittable	Directed/Undirected	$3/2 - \epsilon$ (Thm. 6)
Confluent	Directed	$\Omega(\log n)$ (Thm. 7)*
Unsplittable/Confluent	Directed/Undirected	No $(\frac{15}{14} - \epsilon, 1 + \alpha)$-Approx. (Thm. 10)*

* Corresponding results also hold for the single-sink Maximum Flow Over Time problem (Thms 8, 9, 11).

Table 2 Upper bounds on approximation ratio for variations of the Fixed-Sink Confluent Flow problem. The first three items are for uncapacitated problems but are included here because they serve as the internal building blocks for the approximation algorithms for the capacitated problems.

Network	Capacity	Objective	Sources	Sinks	UB on Approx. Ratio
Static	Uncapacitated	Min Congestion*	n	k $(k \leq n)$	$O(\log^3 n)$ [3]†
Static	Uncapacitated	Min Congestion*	n	k $(k \leq n)$	$1 + \ln k$ [2]
Static	Uncapacitated	Min Congestion*	κ $(\kappa \leq n)$	k $(k \leq n)$	$O(\log^3 \kappa)$ (Thm. 13)†
Static	Node	Max Demand	n	1	$O(\log^6 n)$ with NBA^4 [22]
Static	Edge/Node	Max Demand	κ $(\kappa \leq n)$	1	$O(\log^{10} \kappa)$ with NBA^4 (Thm. 22)†
Dynamic	Edge	Max Flow Over Time	κ $(\kappa \leq n)$	1	$(O(\log^2 \kappa), O(\log^8 n))$ (Thm. 21)†
Dynamic	Edge	Quickest Flow	κ $(\kappa \leq n)$	1	$(O(\log^8 n), O(\log^2 \kappa))$ (Thm. 20)†

* Minimize the maximum node congestion in a network that admits a feasible splittable flow satisfying all supplies.

† These results hold with high probability, or more precisely, with probability $1 - n^{-c}$, where c is a constant.

Section 3.2 proves, for the confluent directed graph case, the much stronger result that unless $P = NP$, it is impossible to construct a polynomial-time $O(\log n)$ approximation algorithm for the single-sink Quickest Flow problem. The major tool used is a modification of a grid graph construction from [21] which was an extension of one pioneered by [8]. We note that our reduction is not the same as that in [21]. There, the objective function was the maximum *amount* of static flow that could be pushed. Here, the objective function is the minimum amount of *time* required to push the supplies. Our proof works by deriving new properties of the grid-graph. Section 3.3 extends the analysis to the Maximum Flow Over Time problem with our lower bounds on the approximation ratio being summarized in Table 1.

We also note that it might seem intuitive that, because confluent flows are "harder" than static flows, the non-approximability of confluent static flows, e.g., the result from [21], should immediately imply the non-approximability of confluent dynamic flows. This is not true, though. The two problems are trying to optimize very different things, making them incomparable. More specifically, in the static case, the goal is *Demand Maximization*, i.e., to find a subset of the demands of maximum total value that can be confluently routed. In the dynamic case, the goal is to find a confluent routing of ALL demands in minimal time. To appreciate the distinction it is instructive to examine confluent routing on *trees* where the static problem is NP-Hard [4] but the dynamic case is *polynomial-time* solvable [17].

Despite the non-approximability shown above for confluent dynamic flows, one might hope to create *bicriteria* (α, β) *approximations*[3]. However, in Section 4, we demonstrate that, for both directed and undirected graphs, there exists a constant $\alpha > 0$ such that, for any $\epsilon > 0$, there is no polynomial-time $(\frac{15}{14} - \epsilon, 1 + \alpha)$-approximation for the Unsplittable/ Confluent Quickest Flow problem, unless $P = NP$. Similar results are obtained for the Unsplittable/Confluent Max Flow Over Time problem. Our proof utilizes a reduction from the Bounded Occurrence 3-Dimensional Matching problem.

In contrast to the above we show, in Section 5, how to construct a $(O(\log^8 n), O(\log^2 \kappa))$-approximation for the Confluent Quickest Flow problem, where κ is now the number of sources, in polynomial time. To this end, we use the idea of routing a confluent flow in a static *monotonic network*, i.e., one in which each vertex is given an additional *vertex capacity* that satisfies that all edges go from a low-capacity node to a high-capacity one, which was introduced in [22]. Recall that in our original confluent flow problem the support of the flow is a tree. In that tree, a parent node never supports less flow than its child. So, intuitively, a feasible confluent flow requires its tree support to be monotonic. We develop new techniques (Theorem 16) that permit constructing, in polynomial time, a confluent flow that routes all supplies in a given monotonic network, while bounding both *node congestion* and *flow length*.

Via this monotonic technique, we build a novel multi-layer monotonic network and construct a confluent static flow on it which is finally re-routed to produce a confluent dynamic flow for our original graph problem. Our method guarantees that a *dynamic flow* can be found such that the total transit time is at most polylogarithmic factor times the optimal. Similarly, this also lets us develop a polynomial-time $(O(\log^2 \kappa), O(\log^8 n))$-approximation of the Confluent Maximum Flow Over Time problem.

Our technique mainly differs from that in [22] in constructing *length-bounded* confluent flows in *static* networks (which might be of independent interest). It also permits us to improve their approximation algorithms when not all vertices are sources. More specifically, recall that [22] gives an $O(\log^6 n)$ approximation algorithm for the demand maximization confluent flow problem, with the no-bottleneck assumption (NBA)[4]. If restricted to static networks, our technique can give an $O(\log^{10} \kappa)$ approximation for the same problem. If κ is bounded, for example, this gives a *constant* approximation, which is nearly optimal.

Our improvement to the approximation ratio comes through a combination of (i) a novel construction of the multi-layer network, and (ii) a new building block inside our monotonic network technique—a better routing approach for *uncapacitated* networks (Theorem 13). This will be discussed in more detail in Section 5.

Our Theorem 13 enables us to route confluent flows in uncapacitated monotonic sub-networks with congestion bounded by $\text{poly}(\log \kappa)$ instead of $\text{poly}(\log n)$. While this might look weak compared to the $1 + \ln k$ (k being the number of sinks) bound from [2] this is only used as a subroutine. In fact, the internal constructions of both [22] and our proofs for approximating the *capacitated* static problem build uncapacitated sub-networks which can have $\Theta(n)$ induced sources and sinks. Plugging in the bound of [2] would give a $\text{poly}(\log n)$ bound. We develop a new combinatorial argument that, combined with our new $\text{poly}(\log \kappa)$ bounds for uncapacitated monotonic sub-networks, gives a $\text{poly}(\log \kappa)$ bound for the capacitated one as well, yielding our Theorem 17. This leads us to the final improvement.

A chart presenting previously known results and our new ones is given in Table 2.

[3] These will be formally introduced in Definition 1.

[4] In node-/edge-capacitated networks, the NBA is that $\max_{v \in V} d(v) \leq \min_{v \in V} c(v)$, and $\max_{v \in V} d(v) \leq \min_{e \in E} c(e)$, respectively.

2 Preliminaries: Definitions and NP-Hard Problems

Let I be some input to an optimization problem, $OPT(I)$ be the *optimum* value to the given problem on I and $|I|$ be its *size*. As examples, I could be a dynamic flow problem on a graph with n vertices and m edges. We could have just as easily defined $|I| = m + n$.

We now define *bicriteria approximation*s for the two-objective optimization problem.

▶ **Definition 1** (Bicriteria Approximation)**.** For any $\alpha, \beta > 0$, an (α, β)-approximation algorithm $\mathcal{A}$ for the two-objective optimization problem is a function that takes as input any parameter k and any instance I, and outputs a solution x such that

1. $\alpha f(x) \geq f(x^*),\ g(x) \leq \beta k$, if the optimization problem is to find a solution x maximizing the cost function $f(x)$ subject to another cost function $g(x) \leq k$,
2. $f(x) \leq \alpha f(x^*),\ \beta g(x) \geq k$, if the optimization problem is to find a solution x minimizing the cost function $f(x)$ subject to another cost function $g(x) \geq k$,

where x^* is the optimal solution for the input I and k.

We can actually define two different types of confluent flows:

▶ **Definition 2.** A flow in G is *node-confluent* if, for every vertex v, all flow leaving v leaves along the same edge. A flow in G is *edge-confluent* if, for every edge $e = (u, v)$ if all flow that passes through e must leave v through the same edge (v, w).

In this paper the term "confluent", when used alone, will denote node-confluence. When edge-confluence is needed (in some proofs) it will be explicitly specified.

Finally we will use the following NP-hard problems in our reductions:

▶ **Definition 3** (The Two-Disjoint Paths (Uncapacitated) Problem)**.** Given a graph G and node pairs $\{x_1, y_1\}$ and $\{x_2, y_2\}$, decide if G contains paths P_1 from x_1 to y_1 and P_2 from x_2 to y_2 such that they are disjoint.

In undirected graphs the Two-Disjoint Paths (Uncapacitated) problem, for both edge-disjoint and node-disjoint paths, is polynomial-time solvable [20]. However, in directed graphs, the problem is NP-hard for both edge-disjoint and node-disjoint paths [6].

▶ **Definition 4** (The Two-Disjoint Paths (Capacitated) Problem)**.** Let G be a (static) graph whose edges are labelled either α or β with $\beta \geq \alpha$. These labels are the *capacities* of the edges. Given node pairs $\{x_1, y_1\}$ and $\{x_2, y_2\}$, decide whether G contains paths P_1 from x_1 to y_1 and P_2 from x_2 to y_2 such that:

i. P_1 and P_2 are disjoint (node-disjoint or edge-disjoint);
ii. P_2 may only use edges of capacity β (P_1 may use both capacity α and capacity β edges).

The version of node-disjoint paths was proven to be NP-hard for undirected graphs by [8]. The version of edge-disjoint paths was proven to be NP-hard by [18].

▶ **Definition 5** (The Bounded Occurrence 3-Dimensional Matching Problem (BO3DM))**.** Suppose there are three disjoint sets $A = \{a_1, ..., a_n\}$, $B = \{b_1, ..., b_n\}$ and $C = \{c_1, .., c_n\}$, and a set $T = \{T_\mu \in A \times B \times C : \mu \in [m]\}$ such that each element of A, B, C occurs in the same constant number M of triples in T. The goal is to find the largest subset $T' \subset T$ such that all triples in T' are disjoint, i.e., no two elements of T' contain the same element of A, B, C.

As shown in [14], there exists an $\epsilon_0 > 0$ such that it is NP-hard to decide whether there exist n disjoint triples in T (*satisfiable* instance) or there exist at most $(1 - \epsilon_0)n$ disjoint triples in T (ϵ_0-*unsatisfied* instance).

Dynamic Flows. We first describe the mechanics of flow over one edge $e = (u, v)$ with capacity c and length ℓ. Suppose there are d units of supply on node u. Assume the discrete case in which d, c, ℓ are all integral and all d need to be moved from u to v. Items move in groups of size at most c, with one group entering e each time unit. Thus, the items are transported in $\lceil d/c \rceil$ groups. It takes ℓ time units for the first group to arrive at v. Since the groups left u at consecutive time units they arrive at v in consecutive time units. Thus, it requires $\lceil d/c \rceil - 1 + \ell$ time to move all items from u to v over e. Also, in both cases, if other items arrived at u wanting to enter e they would have to wait until all items already at u had departed before entering e.

Finally, we introduce some notations. A flow f is *feasible* if $\forall e \in E$, $f(e) \leq c(e)$. For any $e \in E$, we define its *edge congestion* as $EC(e) := f(e)/c(e)$. Under certain circumstance, we may introduce the *node capacity* $c(v)$ of $v \in V$, and define its *node congestion* $NC(v) := f^{out}(v)/c(v)$, where $f^{out}(v)$ is the total flow out of v. For a flow f, we let its edge congestion $EC(f) := \max_{e \in E} EC(e)$ and node congestion $NC(f) := \max_{v \in V \setminus \{t_1,...,t_k\}} NC(v)$, where $t_1, ..., t_k$ are sinks.

A static flow f can be specified by a collection of source-sink paths $\mathcal{P} = (P_1, ..., P_K)$ and corresponding flow values $f_1, ..., f_K$. We define the *length* of flow f as $L(f) := \max_{i \in [k]} L(P_i)$, where $L(P_i) := \sum_{e \in P_i} \ell(e)$ is the length of P_i. f is called as *L-length-bounded* for some $L \in \mathbb{R}^+$ if $L(f) \leq L$, i.e., no path in $\mathcal{P}$ has path length longer than L. Also, if all f_i's are identical, we call f as *uniform*.

3 Approximation Hardness for Unsplittable/Confluent Dynamic Flows

3.1 Constant Approximation Hardness of the Quickest Flows Problem

This section gives a simple proof that a polynomial-time constant approximation algorithm for the single-sink Unsplittable/Confluent Quickest Flow problem would imply $P = NP$.

▶ **Theorem 6.** *The single-sink Unsplittable/Confluent Quickest Flow problem in both directed and undirected graphs cannot be approximated to within a factor* $3/2 - \epsilon$*, for any* $\epsilon > 0$*, unless* $P = NP$.

3.2 Logarithmic Approximation Hardness of Confluent Quickest Flows

For the single-sink directed Confluent Quickest Flow problem we now derive a much stronger result than in the previous section. That is, it is NP-hard to even get a $O(\log n)$ approximation to the optimal solution.

To prove the logarithmic approximation hardness, we construct the following instance.

Hard instance. Before building the desired hard instance, we describe the dynamic half-grid network G_N. It can be viewed as an extension of the static half-grid graph in [21]. There are N rows (numbered from bottom to top) and N columns (numbered from right to left). All the edges in the i-th row and all the edges in the i-th column have capacity $1/i$. The i-th row extends as far as the i-th column and vice versa. The sink t, located at the bottom of the half-grid, is connected with the bottom node t_i of the i-th column by an edge of capacity $1/i$. Also, at the leftmost node of the i-th row, there is a source s_i with supply M^2/i, where M is a sufficiently large constant. We set all edge lengths as 1, and always enforce edge directions to be downwards and to the right.

Suppose we are now given an instance $\mathcal{I}$ of the directed node-disjoint version of the Two-Disjoint Paths (Uncapacitated) problem. We replace each 4-degree node in the half-grid

by a copy of $\mathcal{I}$. Inside the copy, all edges have length 1. Consider the copy of $\mathcal{I}$ at the intersection of the i-th column and j-th row (with $j > i$) in G_N. That instance is incident to two edges of capacity $1/i$ and two edges of capacity $1/j$. Inside that $\mathcal{I}$, we let the edges of capacity $1/j$ be incident to x_1 and y_1, and the edges of capacity $1/i$ be incident to x_2 and y_2; we set all edge capacities to $1/i$. This completes the hard instance of directed confluent dynamic flows. Denote the constructed network as $\mathcal{G}$.

Utilizing $\mathcal{G}$, we obtain the logarithmic approximation hardness for the Confluent Quickest Flow problem. The proof works by showing that if we could get a logarithmic approximation, we could solve $\mathcal{I}$.

▶ **Theorem 7.** *The single-sink Confluent Quickest Flow problem in directed graphs cannot be approximated to a factor within $O(\log n)$, unless $P = NP$.*

3.3 Approximation Hardness of the Max Flow Over Time Problem

This section discusses the approximation hardness of the single-sink Unsplittable and Confluent Maximum Flow Over Time problem.

To derive the approximation hardness of the Unsplittable Maximum Flow Over Time problem, we will again reduce from the directed/undirected edge-disjoint version of Two-Disjoint Paths (Capacitated) problem. We construct the same network as in Section 3.1 and utilizing this constructed network, we show

▶ **Theorem 8.** *The single-sink Unsplittable Maximum Flow Over Time problem in both directed and undirected graphs cannot be approximated to a factor within $3/2 - \epsilon$, for any $\epsilon > 0$, unless $P = NP$.*

Although the above hard instance applies to the confluent flow, we present a stronger lower bound for the Confluent Maximum Flow Over Time in directed graphs.

▶ **Theorem 9.** *The single-sink Confluent Maximum Flow Over Time problem in directed graphs cannot be approximated to a factor within $O(\log n)$, unless $P = NP$.*

4 Constant Bicriteria Approximation Hardness of Dynamic Flows

This section first proves the NP-hardness of constant bicriteria approximations for the Unsplittable and Confluent Maximum Flow Over Time problems.

Our proof uses reductions from the BO3DM problem. Inspired by the reduction[5] presented in [8, 16], given an instance of BO3DM, we construct the following corresponding hard instance for the Unsplittable/Confluent Maximum Flow Over Time problem in undirected graphs. Note that the directed case is similar, except that we enforce all edge directions to point right. Suppose we are given an instance $\mathcal{I}$ of Bounded Occurrence 3-Dimensional Matching problem. Denote the μ-th triple T_μ as $(a_{p_\mu}, b_{q_\mu}, c_{r_\mu})$, where $p_\mu, q_\mu, r_\mu \in [n]$. We build an

[5] Even though we are reducing to the same problem note that our goal differs from [8], which aims at finding a maximum number of length-bounded edge-disjoint paths. For technical reasons, this requires us to develop a totally different bounding technique.

undirected graph $G = (V, E)$ where

$$\begin{aligned} V &= \{s, t\} \cup \{a_{il} : i \in [n], l \in [M-1]\} \cup \{s_i, b_i, c_i : i \in [n]\} \cup \{s'_\mu, x_\mu, y_\mu : \mu \in [m]\}, \\ E &= \{(s_i, s), (s, b_i), (c_i, t), (a_{il}, t) : i \in [n], l \in [M-1]\} \\ &\quad \cup \{(s'_\mu, s), (s, x_\mu), (y_\mu, a_{p_\mu l}) : \mu \in [m], l \in [M-1]\} \\ &\quad \cup \{(b_{q_\mu}, x_\mu), (x_\mu, y_\mu), (y_\mu, c_{r_\mu}) : \mu \in [m]\}. \end{aligned}$$

Hereby, G contains a vertex representing each element in the sets B and C, and $(M-1)$ copies of each element in A. Also, G contains a sink t, and sources s_i ($i \in [n]$), s'_μ ($\mu \in [m]$) as well as one more node s (s is removed when considering confluent flows). Meanwhile, for each triple T_μ in T, there are two vertices x_μ, y_μ to represent it. We connect s_i with s, and s with b_i for each $i \in [n]$; we also connect s'_μ with s, and s with x_μ for each $\mu \in [m]$. Similarly, we connect t with a_{il}, c_i for each $i \in [n]$ and $l \in [M-1]$. For each tuple, $T_\mu = (a_{p_\mu}, b_{q_\mu}, c_{r_\mu})$, we connect x_μ with b_{q_μ}, and y_μ with c_{r_μ} as well as $(M-1)$ copies of a_{p_μ}.

Edge capacities and lengths. All edge capacities are set as 1. Let each (s'_μ, x_μ) have length 5 (red edges), and each (y_μ, c_{r_μ}) have length 4 (green edges), and each (a_{il}, t) have length 3 (blue edges), and all other edges have length 2 (black edges). Finally, we set the time horizon $T = 14$ in the constructed graphs for the Unsplittable/Confluent Maximum Flow Over Time problems. Based on the constructed instance, we have

▶ **Theorem 10.** *There exists a constant $\alpha > 0$ such that, for any $\epsilon > 0$, there is no polynomial-time $(1+\alpha, \frac{15}{14} - \epsilon)$-approximation for the Unsplittable/Confluent Maximum Flow Over Time problem in both directed and undirected graphs, unless $P = NP$.*

To show the hardness of the Unsplittable/Confluent Quickest Flow problem, we construct an instance similar to Theorem 10, except that we let each source have supply 1, and have

▶ **Theorem 11.** *There exists a constant $\alpha > 0$ such that, for any $\epsilon > 0$, there is no polynomial-time $(\frac{15}{14} - \epsilon, 1+\alpha)$-approximation for the Unsplittable/Confluent Quickest Flow problem in both directed and undirected graphs, unless $P = NP$.*

5 Polylogarithmic Approximation for Confluent Dynamic Flows

5.1 Static Confluent Flows in Uncapacitated Networks with κ Sources

We now develop techniques for routing confluent flows in uncapacitated networks with $\kappa \leq n$ sources. Through Section 5.3, unless otherwise specified, the flow discussed is *static.*

▶ **Definition 12** (β-Satisfiable)**.** For any $\beta \in [0, 1]$, a supply d_i is *β-satisfiable* in flow f if at least a β faction of d_i can be sent to the sink via f. A flow f is β-satisfiable if all supplies are β-satisfiable in f.

Again, suppose $G = (V, A)$ is a static directed graph with supply $d(v)$ located at each $v \in V$. There exists a collection of sinks $\{t_1, ..., t_k\} \subset V$. We let κ be the number of non-zero supplies, and let all edge and node capacities be 1. We present

▶ **Theorem 13.** *In the directed uncapacitated network with κ uniform non-zero supplies, given a (splittable) 1-satisfiable flow f, there exists a randomized algorithm for finding a multi-sink confluent flow f' with the node congestion bounded by $O((NC(f))^2 \log^3 \kappa)$ whp*[6]*.*

[6] Throughout the paper, we use *whp* to mean with high probability, or more precisely, with probability $1 - n^{-c}$, where n is the number of nodes in the network and c is a constant.

Note that if κ is bounded and f is feasible, Theorem 13 can provide confluent flows with constant congestion.

Also, the support of the resulting flow is a collection of trees rooting at those sinks $t_1, ..., t_k$. We guarantee that the height of those trees can be bounded as below.

▶ **Lemma 14.** *Whp, the height of any tree constructed in the randomized algorithm is at most* $O(NC(f) \log n)$.

5.2 Static Length-Bounded Confluent Flows in Monotonic Networks

This section gives an algorithm for constructing a length-bounded confluent flow in *monotonic networks*, utilizing techniques developed in Section 5.1. A monotonic network is a special (static) directed graph with vertex capacities and no edges pointing in the direction of decreasing capacity. Formally,

▶ **Definition 15** (Monotonic Network)**.** A directed graph $G = (V, A)$ with node capacity $c(v)$ for each $v \in V$ is a *monotonic network* iff $c(u) \leq c(v)$ for every arc (u, v).

The network $G = (V, A)$ is the same as Section 5.1 except that here each node has capacity $c(v)$ and each edge has capacity 1. Our first step is to prove

▶ **Theorem 16.** *Let* $G = (V, A)$ *be a monotone network. Given a* 1*-satisfiable flow* f *with node congestion at most 1, one can, in polynomial time, construct a confluent* 1*-satisfiable flow with node congestion* $O(\log^8 n)$ *and flow length* $O\left(L(f) \log n \log c_{\max} / \log \log n\right)$ *whp, even without the no-bottleneck assumption.*

The idea is to first decompose the monotonic network into several sub-networks, and in each, construct length-bounded confluent flows with small node congestion. Connecting all confluent flows in those sub-networks, we can construct a confluent flow in the original network as desired. Our monotonic network technique incorporates a new parameter, namely the edge length, and, more importantly, our objective is to construct a *bicriteria* confluent flow, namely bounding *both* node congestion and length (note that in [22], only node congestion can be bounded). The main difference from [22] lies in that we embed our new algorithms for uncapacitated networks into the monotonic network routing.

Our technique can be further improved if we remove the length-bounded constraint. The key observation is that the sources in each sub-network are only induced by the given (splittable) flow that we would like to re-route into a confluent one. We can guarantee that, if the given flow is unsplittable, at most κ flow paths pass between two sequential sub-networks, inducing at most $O(\kappa)$ sources. This, combined with our new technique for uncapacitated networks, gives the improvement of the congestion from $\text{poly}(\log n)$ to $\text{poly}(\log \kappa)$.

▶ **Theorem 17.** *Let* $G = (V, A)$ *be a monotone network with a single sink. If there is* 1*-satisfiable flow* f *with node congestion at most 1, one can, in polynomial time, construct a confluent* 1*-satisfiable flow with node congestion* $O(\log^8 \kappa)$ *whp, under the NBA.*

5.3 Static Length-Bounded Confluent Flows in General Networks

Via the techniques developed above for monotonic networks, this section develops a polynomial-time algorithm for determining a length-bounded confluent static flow in general networks.

Suppose we are given a directed/undirected *edge-capacitated* network $G(V, E)$ (Section 5.2 dealt with *node* capacitated networks). Each node $v \in V$ has a supply $d(v)$ to be sent to

the unique sink t. Our goal is to find *a subset of supplies* of maximum total value that can be routed via a confluent flow, whose flow length and edge congestion are both bounded.

To this end, we need to pre-process the network as follows. First, we ignore those demands of size at most $d_{\max}/2\kappa$, as they contribute at most half of the value of the optimal flow. Meanwhile, we round each supply up to the nearest power of 2, and group those with the same value together, producing $O(\log \kappa)$ groups of distinct supply sizes. To compute an approximation, we will separately route each supply group in G, and output the flow of the maximum value among all groups. Note that, this will lose a $O(\log \kappa)$ factor in the approximation ratio. Hence, we reduce the original problem to the uniform-supply case. Without loss of generality, by scaling, we can assume every supply is 1.

Second, we round each capacity up to the nearest power of 2, and assume all edges have capacity at most $\kappa d_{\max}$, i.e., $c_{\max} \leq \kappa d_{\max}$ as the extra capacity above this value is superfluous. Furthermore, when considering the uniform-supply case, those edges with capacity less than the supply size would never be used, as the supply should be routed confluently. Accordingly, we can assume each edge capacity is in $[1, \kappa]$ as $d_{\max} = 1$ in unit-supply case, and then there exist $O(\log \kappa)$ distinct capacity sizes.

Given a directed/undirected edge-capacitated network $G(V, A)$ with a single sink t, letting $k := \lfloor \log c_{\max} \rfloor + 1$, we construct the directed *k-layer (monotonic) network* H.

- ***k* layers.** Create k layers and k node sets $V(H_0), V(H_1), ..., V(H_{k-1})$, where $V(H_i) := V(G) \setminus \{t\}$ and the i-th layer contains $V(H_i)$.
- **Induced node capacities.** For the i-th node set $V(H_i)$ $(i = 0, ..., k-1)$, denote by u^i the i-th copy of node u, and let u^i have capacity 2^i.
- **Vertical arcs.** For each edge $(u, v) \in A(G)$, connect two vertical arcs (u^i, v^i) (and (v^i, u^i) if G is undirected) with capacity of 2^i in H, iff the capacity of (u, v) is at least 2^i $(i = 0, ..., k-1)$.
- **Horizontal arcs.** For $0 \leq i \leq k-2$, $\forall u \in V$, connect a horizontal arc (u^i, u^{i+1}) with capacity 2^i.
- **Arc lengths.** Let vertical arcs have the same length as arcs in G, and horizontal arcs have length 0.
- ***H* := (*V*(*H*), *A*(*H*)).** Set $V(H)$ as the union of $V(H_0), V(H_1), ..., V(H_{k-1}), \{t\}$ plus those dummy sinks, and set $A(H)$ as the collection of those vertical and horizontal arcs.
- **Supplies.** Place the supply of v at its copy v^0 in Layer 0.
- **Dummy sinks.** If there exists an edge (u, t) with capacity of 2^i, then create a copy t_u^j of t in Layer j and let the capacity of t_u^j be 2^j, for each $j = i, ..., k-1$. Connect the vertical arc (u, t_u^i) with capacity of 2^i, and the horizontal arc (t_u^j, t_u^{j+1}) with capacity of 2^j, for each $j = i, ..., k-2$. Finally, connect the arc (t_u^{k-1}, t) with capacity of 2^{k-1}.

Our multi-layer network can be viewed as a new construction enabling our length-bounded routing technique to work in edge-capacitated networks. Applying Theorem 16 yields:

▶ **Theorem 18.** *In the layered network H, given a (splittable) flow f for routing all unit supplies with node congestion at most 1, there exists a polynomial-time algorithm for constructing a 1-satisfiable confluent flow with node congestion $O(\log^8 n)$ and flow length $O\left(L \log^2 n / \log \log n\right)$ whp.*

Thus, via Theorem 18, we can obtain a confluent flow h in the k-layer network H with both node congestion and length being bounded. Nevertheless, since H is constructed from logarithmic copies of nodes in G, the constructed confluent flow h in H may induce a non-confluent flow in G, because some vertices v might contain logarithmic out-flow edges. We then show that there is a polynomial-time scheme for re-routing h into a confluent flow

in the original network G. Also, although we bound *node* congestion in H, the original network G is in fact *edge*-capacitated and we are actually interested in the edge congestion. Fortunately, our construction of multi-layer networks can be patched. With the help of the monotonic structure and dummy sinks, we can bound the edge congestion.

Combining everything, we conclude that

▶ **Theorem 19.** *Suppose G is a directed/undirected edge-capacitated network with one sink. If there is an L-length-bounded confluent flow for routing all supplies with edge congestion at most 1 in G, then, there exists a polynomial-time algorithm for finding a confluent flow for routing* a subset of supplies *with value at least $\sum_{i\in[\kappa]} d_i/O(\log^2 \kappa)$, with edge congestion $O(\log^8 n)$ and flow length $O(L \cdot \log^3 n/\log\log n)$ whp.*

5.4 Polylogarithmic Approximation for the Confluent Dynamic Flows

With the techniques developed the polylogarithmic approximation for the confluent dynamic problem can be shown to immediately follow. We do not use any storage at intermediate nodes.

▶ **Theorem 20.** *In directed/undirected, edge-capacitated dynamic networks, there is a polynomial-time algorithm that constructs an $(O(\log^8 n), O(\log^2 \kappa))$-approximation for the single-sink Confluent Quickest Flow problem whp.*

▶ **Theorem 21.** *In directed/undirected, edge-capacitated dynamic networks, there is a polynomial-time algorithm that constructs an $(O(\log^2 \kappa), O(\log^8 n))$-approximation for the single-sink Confluent Maximum Flow Over Time problem whp.*

Our technique can be restricted to static flows, yielding

▶ **Theorem 22.** *In directed/undirected, edge-/node-capacitated static networks that satisfy the no-bottleneck assumption, there is a polynomial-time algorithm that constructs an $O(\log^{10} \kappa)$-approximation for the single-sink Demand Maximization Confluent Flow problem whp.*

Acknowledgement. We would like to thank the authors of [22] for providing us with a pre-print of the full version of their paper.

References

1 A. Bley. Routing and capacity optimization for IP networks. In *Operations Research Proceedings 2007*, pages 9–16. Springer, 2008.

2 J. Chen, R. D. Kleinberg, L. Lovász, R. Rajaraman, R. Sundaram, and A. Vetta. (Almost) tight bounds and existence theorems for single-commodity confluent flows. *Journal of the ACM*, 54(4):16, 2007.

3 J. Chen, R. Rajaraman, and R. Sundaram. Meet and merge: Approximation algorithms for confluent flows. In *Proceedings of STOC'03*, pages 373–382. ACM, 2003.

4 D. Dressler and M. Strehler. Polynomial-time algorithms for special cases of the maximum confluent flow problem. *Discrete Applied Mathematics*, 163, Part 2:142–154, 2014.

5 L. R. Ford and D. R. Fulkerson. Constructing Maximal Dynamic Flows from Static Flows. *Operations Research*, 6(3):419–433, jun 1958.

6 S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

7 M. Golin, H. Khodabande, and B. Qin. Non-approximability and polylogarithmic approximations of the single-sink unsplittable and confluent dynamic flow problems (full version). *arXiv*, 2017. `arXiv:1709.10307`.

8 V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67(3):473–496, 2003.

9 A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379(3):387–404, 2007.

10 D. G. Harris and A. Srinivasan. Constraint satisfaction, packet routing, and the Lovasz Local Lemma. In *Proceedings of STOC'13*, pages 685–694, New York, NY, USA, 2013. ACM.

11 B. Hoppe and É. Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of SODA'94*, pages 433–441, 1994.

12 B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000.

13 N. Kamiyama. *Studies on Quickest Flow Problems in Dynamic Networks and Arborescence Problems in Directed Graphs*. PhD thesis, Kyoto University, 2009.

14 V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35, 1991.

15 E. Köhler, R.H. Möhring, and M. Skutella. Traffic networks and flows over time. In *Algorithmics of Large and Complex Networks*, pages 166–196. Springer, 2009.

16 S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proceedings of FOCS'97*, pages 426–436. IEEE, 1997.

17 S. Mamada, T. Uno, K. Makino, and S. Fujishige. A tree partitioning problem arising from an evacuation problem in tree dynamic networks. *Journal of the Operations Research Society of Japan*, 48(3):196–206, 2005.

18 G. Naves, N. Sonnerat, and A. Vetta. Maximum flows on disjoint paths. In *Approximation, Randomization, and Combinatorial Optimization*, pages 326–337. Springer, 2010.

19 M. M. B. Pascoal, M. E. V. Captivo, and J. C. N. Clímaco. A comprehensive survey on the quickest path problem. *Annals of Operations Research*, 147(1):5–21, aug 2006.

20 N. Robertson and P.D. Seymour. Graph minors .XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

21 F. B. Shepherd and A. Vetta. The inapproximability of maximum single-sink unsplittable, priority and confluent flow problems. *ArXiv*, abs/1504.00627, 2015.

22 F. B. Shepherd, A. Vetta, and G. T. Wilfong. Polylogarithmic approximations for the capacitated single-sink confluent flow problem. In *Proceedings of FOCS'15*, pages 748–758, 2015.

23 Martin Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization*, pages 451–482. Springer, 2009.