Bursting Flow Query on Large Temporal Flow Networks

Abstract

Recently, queries that find bursting patterns in temporal graph data have received increasing research attention. In particular, finding the flow in temporal networks whose flow values are bursting in a time interval has numerous applications, such as detecting the money laundering by the maximum average transfer flow in a transaction graph, and the congestion by the maximum average traffic flow in a road network. Despite its usefulness, there is limited research on querying such a flow pattern. In this paper, we study a novel query of finding *a flow pattern of burstiness* in a temporal flow network. In a nutshell, this query aims to find the *bursting flow* f from a source node to a sink node such that the ratio of f's flow value to the time interval length of f is maximized. To solve this query, we propose the first solution called BFQ that enumerates all the necessary time intervals and then computes the maximum flow value for each interval. Based on BFQ, we propose an efficient solution called BFQ*, which consists of optimization techniques that incrementally compute the maximum flows without computing the common parts of flows from scratch. The experimental results demonstrate the efficiency of our solutions. A case study on a real world transaction network demonstrates the application of this bursting flow query on detecting abnormal transactions.

1 Introduction

Temporal graphs, such as social networks, blockchain networks and road networks [40, 45, 48, 50], have emerged widely in many applications, including social network analysis, transportation routing and crime detection [15, 23, 26, 29, 30, 38, 52]. In a temporal graph, an edge can be denoted by a triple (u, v, τ), where u and v are two nodes, and τ represents the timestamp of the interaction between u and v [3, 22]. Moreover, the edges may have their capacities and in such case, the temporal networks are more accurately referred to as *temporal flow networks*. As the interaction patterns in temporal graphs are often known to be *bursty* (*i.e.*, the interaction events of humans often happen in a short time interval), the bursting queries are emerging in temporal graph applications [7, 9, 10, 33, 54].

In this paper, we study *a novel flow problem on temporal flow networks* called δ-bursting flow (δ-BFlow). Given a source node s and a sink node t of a temporal flow network N_T, the δ-BFlow problem is to find in N_T a temporal flow F starting from s at starting timestamp τ_s to t at ending timestamp τ_e, such that i) the length of F's time interval $[\tau_s, \tau_e]$, *i.e.*, $\tau_e - \tau_s$, is not less than the parameter δ, and ii) the flow density of F, defined by the ratio of F's flow value (*e.g.*, the total transaction amount from s to t during $[\tau_s, \tau_e]$ in transaction networks) to the length of F's time interval, is maximized. We note that, for a temporal flow having a tiny time interval such as a one-off transaction, its flow density may become enormous that this density does not indicate non-trivial bursting patterns. Therefore, we propose to use the parameter δ for filtering these flows. We illustrate the applications of the δ-BFlow query with an example.

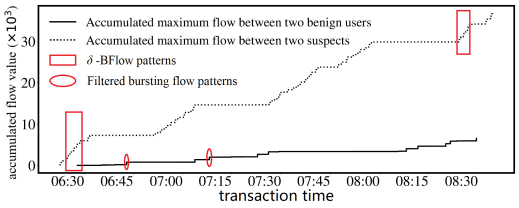


Figure 1: Examples of δ-BFlows on a transaction network

Example 1. *A digital payment company recently finds two fraud suspects, and wants to trace whether a large volume of money was transferred from one to another in a short span of time. The transfers may not be direct transfers but through a large network of people. The company wants to look for a time interval where the density of money transferred was the most, i.e., the amount of money transferred divided by the length of the transfer time interval. It would also like to filter any trivial flows due to one-off transactions, so it would like the transfer time interval having a length at least δ. The transaction network can be formatted as a temporal flow network satisfying the following: for any edge e from node u to node v at timestamp τ, e is associated with a capacity equal to the transaction amount transferred from user u to user v at τ. Then, Figure 1 shows by (red) rectangles some examples of such bursting flow pattern between these two suspects in this temporal flow network found by a δ-BFlow query, where δ refers to 4 minutes. Note that there are also bursting flows shown by (red) ellipses between benign users whose time intervals are relatively short (e.g., 6:48-6:49 and 7:13-7:14), while δ can be used to filter these patterns. Similar scenarios of finding the bursting flow pattern can be found in many other applications, such as [4, 11, 31, 41].*

There have been works on finding the maximum flow in temporal flow networks [2, 19, 27], which focus on finding the absolute maximization of flow values. However, such maximization on temporal flow may happen during a long time interval, which cannot quantify the speed of temporal flows for finding the δ-BFlow. Hence, in this paper, we take the first step towards querying the temporal flows on temporal flow networks that have the *maximum average flow value during their time intervals*. There are two main challenges.

Challenge 1: To define the bursting flow in temporal flow networks.

To define a novel notion of bursting flow in temporal flow networks, we take as the metrics of the burstiness the average flow value of a flow f from source node s to sink node t during f's time interval from f's starting timestamp on s to f's ending timestamp on t. In particular, we propose a novel concept of a bursting flow in temporal flow networks called δ-bursting flow (δ-BFlow), whose time interval has a length of at least a parameter δ. The δ value is used to filter trivial bursting flows that happen during extremely short time intervals since the average flow value can be large due to a tiny time interval even when this flow has a small flow value.

Challenge 2: To propose efficient solution for finding δ-BFlow.

To find the novel δ-BFlow in temporal flow networks, existing methods cannot be applied directly since either the solutions for other

SIGMOD'2025, June 22-27, 2025, Berlin, Germany

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

bursting pattern queries are not compatible with the flow computation, or the solutions for discovering classical flow patterns on flow networks cannot handle the burstiness on temporal flow networks. Therefore, to tackle this challenge, we first propose a practical solution to find δ-BFlow in a temporal flow network N_T by two steps: i) enumerate all possible candidate time intervals of N_T; and ii) transform N_T into flow networks *w.r.t.* these candidate intervals and then find the maximum flows (Maxflows) in these transformed flow networks. To further improve the efficiency of this solution, we propose optimized algorithms that incrementally compute Maxflows for the candidate intervals instead of finding them from scratch.

To the best of our knowledge, this is *the first work to study the bursting flow problem on temporal flow networks*.

Contributions. The contributions of this paper are as follows.

- We propose a novel concept of a bursting flow in the temporal flow network called *δ-bursting flow* (δ-BFlow) to capture the pattern of temporal flows whose flows burst in a certain time interval. δ-BFlow is a flow from the source node to the sink node that has the maximum flow density, that is the ratio of its flow value to the length l of its time interval where l is not less than a parameter δ. In this work, we study a *novel* δ-BFlow *query* that is to find the flow density and the time interval of δ-BFlow from a given source node to a given sink node in a temporal flow network. The δ-BFlow query can be answered in polynomial time.
- We propose the first practical solution called BFQ for answering the δ-BFlow query. BFQ has two procedures: i) transforming a temporal flow network N_T into an equivalent flow network for each of the $O(d^2)$ enumerated candidate time intervals instead of all the $O(|T|^2)$ possible time intervals, where d and $|T|$ are the degree of the source node (or the sink node) and the number of timestamps in N_T, respectively, and ii) computing the flow value $|f|$ of Maxflow f in the transformed flow network for each interval $[\tau_s, \tau_e]$ and hence, the flow density $|f|/(\tau_e - \tau_s)$ by using the traditional Maxflow algorithms.
- To further optimize the efficiency of BFQ, we propose incremental Maxflow algorithms to avoid the redundant computation on finding the same augmenting paths when computing Maxflows on the transformed flow networks for different time intervals. We also propose pruning strategies to accelerate the Maxflow computation.
- We conduct comprehensive experiments on real world temporal flow networks to evaluate the efficiency of our proposed algorithms. A case study on a real world transaction network is presented to demonstrate the application of δ-BFlow for anomaly detection.

Organization. Section 2 discusses the related work. Section 3 presents the preliminaries and the problem statement. A practical solution is proposed in Section 4. Section 5 presents our efficient optimized solution. Experimental evaluations are presented in Section 6. Section 7 concludes the paper and presents future works.

2 Related Work

There have been a lot of research studying the maximum flow problem on traditional networks [12, 13, 17] due to numerous applications in scientific and engineering computing. In recent years, analysis on temporal graph query processing has attracted much attention and there comes various temporal query processing [15, 29, 32].

Maximum flow problem (Maxflow **problem).** Maxflow problem is a classical problem. Various solutions have been proposed over the past few decades. One kind of solutions makes use of the *augmenting path*, which is a path of positive capacity from a source node to a sink node. Ford et al. [13] proposed the first algorithm to answer the Maxflow problem by iteratively finding an augmenting path in the network. As a practical solution which is also used in this work, Dinic [12] proposed to i) construct a layered network called *level graph* from the residual network of the original network by using the *breadth-first search*, and ii) find an extension of the shortest augmenting path called *blocking flow* in the level graph. These steps are repeated till no more blocking flows can be found. Hochbaum [20] proposed the *pseudoflow* algorithm to organize all unsaturated edges in the flow network by using a normalized tree and then accelerate the process of finding augmenting paths by incorporating the dynamic tree [36].

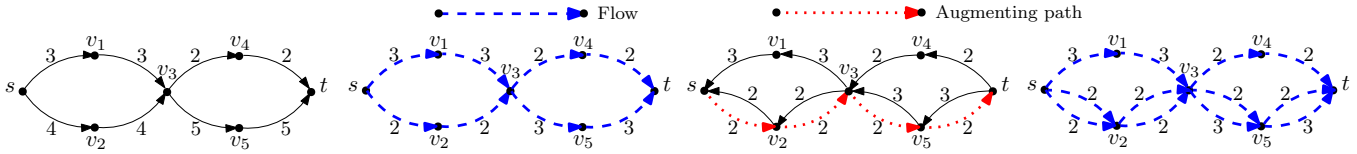
Different from augmenting path based methods, Goldberg et al. [17] proposed a *push-relabel* method to compute the maximum flow, which consists of i) the push operation that move units of flow from an active node to its neighbors, and ii) the relabel operation that updates the labels of the active nodes based on the labels of the neighbor nodes. Both operations are repeatedly performed till no more applicable operations can be performed. There are also works on finding Maxflow on networks of specific topologies with more efficient solutions [5, 21]. Recently, Chen et al. [6] proposed a Maxflow algorithm with a theoretically almost-linear time complexity. However, it has no practical implementation.

As a widely used Maxflow algorithm,[1] Dinic takes $O(|V|^2 \cdot |E|)$ time, where V and E are the set of nodes and the set of edges in a flow network, respectively. The time complexities of some other existing Maxflow algorithms are summaried in Appendix A in the technical report [1].

Query processing on the temporal network. There has been a growing interest in query processing on temporal graphs, such as reachability queries [47, 55], path queries [38, 39, 46, 53], subgraph queries [9, 24, 54, 56] and community search queries [29, 32, 51]. Recently, Chen et al. [8] studied the reachability queries on temporal bipartite graphs; Wen et al. [42, 43] studied the span-reachability queries on large temporal graphs by relaxing the time order dependency of paths; Huang et al. [25] studied the minimum path pair query problem of the time-varying network; Qin et al. [33] studied mining bursting core in large temporal graphs; and Xie et al. [49] studied querying connected components in large temporal graphs *without* considering the flow within the components.

Regarding the flow problem on temporal networks, [14, 34, 44] studied the earliest arrival flow problem to determine the earliest time that a flow comes from a source node to a sink node. To solve Maxflow on dynamic networks with an insertion or deletion of an edge, Kumar et al. [28] proposed an incremental algorithm based on the push-relabeling algorithm while Greco et al. [18] proposed incremental algorithm based on finding the augmenting paths. However, these methods cannot be adopted directly on finding temporal flows due to the time constraint on the edges of temporal paths. To study Maxflow on temporal networks, Horst et al. [19] assumed

[1] Dinic often has the best overall performance in practice due to many efforts on implementation optimizations.



(a) Flow network N=(V, E, C), where the edge denotes C:$V\times V\rightarrow\mathbb{R}_{\geq 0}$ (b) Flow f from s to t in N, where the edge denotes f:$V\times V\rightarrow\mathbb{R}_{\geq 0}$ (c) The residual network N_f=(V, E_f, C_f), where the edge denotes C_f:$V\times V\rightarrow\mathbb{R}_{\geq 0}$ (d) Maxflow f' from s to t in N, where the edge denotes f':$V\times V\rightarrow\mathbb{R}_{\geq 0}$

Figure 2: Illustration of the notations in classical flow networks

that the capacity of each edge changes over time while Akrida et al. [2] characterized the edges in the temporal flow network with specific or random availabilities on time. Kosyfaki et al. [27] studied the Maxflow problem on temporal flow networks with the classical flow transfer model and a greedy flow transfer model that the maximum possible quantity of capacity is transferred on each edge.

Recently, the bursting queries are emerging in temporal graph applications [7, 9, 10, 33, 54]. To the best of our knowledge, the δ-bursting flow problem has not been studied before, for which existing solutions for answering other pattern queries in temporal graphs [7, 9, 10, 49, 54] cannot be adopted. The reason is that these queries do not imply flow. In particular, [33] studied the mining of bursting core in large temporal graphs. However, there can be bursting flows in a non-core subgraph, whereas there can be bursting cores with small flow values. Although the solution of [27] adopted the linear programming (*LP*) to compute Maxflows in temporal flow networks, it is reported in its experimental results that *LP* cannot handle temporal networks with more than 10K edges for computing Maxflows efficiently.

3 Preliminaries

In this section, we present notations for describing the bursting flow problem on temporal flow networks.

3.1 Background

We first introduce some definitions for classical flow problems.

Flow network. A *flow network*, denoted by $N = (V, E, C)$, is a directed graph, where i) V and E are the sets of nodes and edges, respectively, and ii) C: $V \times V \rightarrow \mathbb{R}_{\geq 0}$ is a *capacity mapping function* such that $\forall\, u, v \in V$, if $(u, v) \in E$, $C(u, v)$ equals a positive value which denotes the capacity of the edge from u to v; otherwise, $C(u, v) = 0$.

Flow. Given a flow network $N = (V, E, C)$, a *flow* from a source node s (or simply source) to a sink node t (or simply sink) is denoted by a mapping function f: $V \times V \rightarrow \mathbb{R}_{\geq 0}$ that satisfies the following:

a) **Capacity constraint.** $\forall u, v \in V$, $0 \leq f(u, v) \leq C(u, v)$; and
b) **Flow conservation.** $\forall u \in V - \{s, t\}$, $\sum_{v\in V} f(v, u) = \sum_{v\in V} f(u, v)$.

For a flow f from source s to sink t, f's *flow value* $|f|$ is,

$$|f| = \sum_{v\in V} f(s, v) = \sum_{v\in V} f(v, t) \tag{1}$$

Then, the maximum flow is defined as follows.

Definition 1. (***Maximum flow*** (Maxflow)) *Given a flow network N, a source s, and a sink t, the* maximum flow *from s to t in N is a flow f from s to t in N such that the flow value $|f|$ is maximized.*

To *compute* Maxflow from a source to a sink in flow networks, the *residual network* and *augmenting path* are used in the classical Maxflow algorithms such as *Ford-Fulkerson* [13] and *Dinic* [12].

Residual network. Given a flow network $N = (V, E, C)$ and a flow f in N, the *residual network* of N *w.r.t.* f, denoted by $N_f = (V, E_f, C_f)$, is a directed graph that satisfies the following:

- V is the set of nodes;
- C_f: $V\times V \rightarrow \mathbb{R}_{\geq 0}$ is a capacity mapping function such that if $(u, v) \in E$, $C_f(u, v) = C(u, v) - f(u, v)$ and $C_f(v, u) = f(u, v)$. Otherwise, $C_f(u, v) = 0$; and
- $E_f = \{(u, v) \mid u, v \in V \text{ and } C_f(u, v) > 0\}$ is the set of edges.

Augmenting path. Given a resident network $N_f = (V, E_f, C_f)$ of a flow network N *w.r.t.* a flow f from source s to sink t in N, an *augmenting path* is a path from s to t in this resident network N_f. For an augmenting path p in N_f, the value of the flow passes through p (or simply the flow value of p), denoted by Flow(p), equals to,

$$\text{Flow}(p) = \min_{(u,v) \text{ is an edge of } p} C_f(u, v) \tag{2}$$

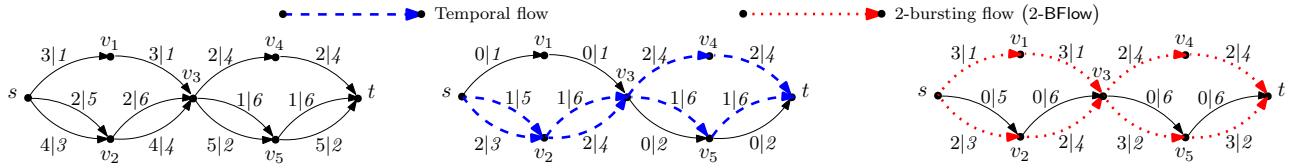
According to the definitions of the residual network and the augmenting path, Flow(p) is *non-zero*. Note that C_f of a residual network is defined based on an assumption on the flow network $N = (V, E, C)$: if there exists an edge from node u to node v in N, there are no edges from v to u in N [17, 37].[2] For simplicity, we apply the same assumption in this paper.

Augmenting path-based Maxflow **algorithm.** Given a source s and a sink t, existing classical augmenting path-based Maxflow algorithms, *Ford-Fulkerson* [13] and *Dinic* [12], compute Maxflow from s to t by iteratively finding augmenting paths from s to t in the residual network *w.r.t.* the currently found flow till no more augmenting paths can be found. In this paper, we adopt the Dinic solution [12] for computing Maxflows,[1] while other augmenting path-based Maxflow algorithms can be also applied in our solutions.

Example 2. *Figure 2 illustrates the notations with examples.*

- ***(Flow network)*** *Figure 2(a) shows an example of a flow network $N = (V, E, C)$, where i) $V = \{s, v_1, v_2, v_3, v_4, v_5, t\}$, ii) $E = \{(s, v_1), (s, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_4), (v_3, v_5), (v_4, t), (v_5, t)\}$, and iii) $C(s, v_1) = C(v_1, v_3) = 3$, $C(s, v_2) = C(v_2, v_3) = 4$, $C(v_3, v_4) = C(v_4, t) = 2$, $C(v_3, v_5) = C(v_5, t) = 5$, and $C(u, v) = 0$, as $(u, v) \notin E$;*
- ***(Flow)*** *The (blue) dashed paths in Figure 2(b) show an example of a flow f from source s to sink t in N, where $f(s, v_1) = f(v_1, v_3) = f(v_3, v_5) = f(v_5, t) = 3$, $f(s, v_2) = f(v_2, v_3) = f(v_3, v_4) = f(v_4, t) = 2$, and $f(u, v) = 0$, as $(u, v) \notin E$. The flow value $|f|$ is 5;*
- ***(Residual network and augmenting path)*** *Figure 2(c) shows the residual network N_f of N w.r.t. flow f in Figure 2(b). The (red)*

[2] To handle the case that both of (u, v) and (v, u) exist in E, we can revise the flow network $N = (V, E, C)$ by removing (v, u) and then creating a new node w and two edges (v, w), (w, u) such that $C(v, w) = C(w, u) = C(v, u)$. In this way, existing Maxflow algorithms by finding augmenting paths in residual networks can be used on the revised network of N to compute Maxflow in N.



(a) Temporal flow network N_T=(V, E_T, T, C_T) where T=[1, 2, 3, 4, 5, 6] and the edge denotes C_T:$V\times V\times T\to\mathbb{R}_{\geq 0}|\tau\in T$

(b) Temporal flow F from s at timestamp 3 to t at timestamp 6 in N_T, where the edge denotes F:$V\times V\times T\to\mathbb{R}_{\geq 0}|\tau\in T$

(c) 2-BFlow F' from s at timestamp 1 to t at timestamp 4 in N_T, where the edge denotes F':$V\times V\times T\to\mathbb{R}_{\geq 0}|\tau\in T$

Figure 3: Illustration of the notations in temporal flow networks

dotted path p: $s\to v_2\to v_3\to v_5\to t$ from s to t in Figure 2(c) shows an augmenting path in N_f having Flow(p) = 2*; and*

- *(*Maxflow ***computation)*** *Figure 2(d) shows in blue a* Maxflow f' *from s to t in N having $|f'|=|f|+$ Flow(p) = 7, which is obtained by finding the augmenting path p in the residual network N_f as shown in Figure 2(c). As no more augmenting paths can be found, f' is a* Maxflow.

To compute Maxflow*s efficiently, we adopt the* Dinic *solution that uses the ideas of* level graph *and* blocking flow *for finding augmenting paths. Due to the page limitation, we only show a simplified procedure of how* Dinic *computes* Maxflow *from s to t in N by iteratively finding augmenting paths.*[3] *First,* Dinic *initializes the current flow f and the residual network N_f w.r.t. f as a mapping function $f: V\times V\to 0$ and the flow network N shown in Figure 2(a), respectively. In the first iteration,* Dinic *i) computes the level graph based on N_f; ii) computes by using this level graph the blocking flow by finding in N_f the augmenting paths p_1: $s\to v_1\to v_3\to v_4\to t$ having* Flow(p_1) = 2*, p_2: $s\to v_1\to v_3\to v_5\to t$ having* Flow(p_2) = 1*, and p_3: $s\to v_2\to v_3\to v_5\to t$ having* Flow(p_2) = 4*; and iii) updates the current flow f by including this blocking flow (i.e., the flow passing through these three augmenting paths). In the second iteration,* Dinic *continues to compute the level graph by the updated N_f. The computed graph indicates that no more augmenting paths can be found in N_f, and hence,* Dinic *stops and outputs a* Maxflow *having a flow value equal to 7.*

3.2 Temporal Flow and Problem Statement

In this subsection, we extend some definitions above in the *temporal network* for self-containedness and consistency of symbols, and then define the *δ-bursting flow*. These symbols are compatible with some existing works [27, 33, 49].

Temporal network. A *temporal network*, denoted by a 3-ary tuple (V, E_T, T), is a directed graph, where i) V is the set of nodes, ii) T = [τ_1, ..., τ_n] is a sequence of timestamps such that $\tau_j-\tau_{j-1}$, $j\in[2,n]$ is a constant (w.l.o.g, the constant is assumed to be 1 unless specified otherwise), and iii) $E_T=\{(u,v,\tau)\mid u,v\in V,\tau\in T\}$ is the set of temporal edges. A temporal edge $e=(u,v,\tau)\in E_T$ denotes a directed edge from node u to node v having timestamp τ.

Temporal flow network. A *temporal flow network*, denoted by $N_T=(V,E_T,T,C_T)$, is a directed graph, where i) (V, E_T, T) is a temporal network, and ii) C_T: $V\times V\times T\to\mathbb{R}_{\geq 0}$ is a capacity mapping function such that $\forall\ u,v\in V$, $\tau\in T$, if $(u,v,\tau)\in E_T$, $C_T(u,v,\tau)$ equals a positive value which denotes the capacity of the temporal edge from u to v having timestamp τ; otherwise, $C_T(u,v,\tau)=0$.

Temporal flow. Given a temporal flow network N_T = (V, E_T, T, C_T), a *temporal flow* from source s at starting timestamp τ_s to sink t at ending timestamp τ_e ($\tau_e>\tau_s$) is denoted by a mapping function F: $V\times V\times T\to\mathbb{R}_{\geq 0}$ that satisfies the following:

a) **Capacity constraint.** $\forall u,v\in V,\tau\in T, 0\leq F(u,v,\tau)\leq C_T(u,v,\tau)$;

b) **Flow conservation.** $\forall u\in V-\{s,t\}$,

$$\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]}F(v,u,\tau)=\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]}F(u,v,\tau)\ ;\ \text{and}\qquad(3)$$

c) **Flow's time constraint.** $\forall u\in V-\{s,t\}$, $\tau'\in[\tau_s,\tau_e]$,

$$\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']}F(v,u,\tau)\geq\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']}F(u,v,\tau)\qquad(4)$$

Intuitively, the time constraint of a temporal flow F enforces that, at any node u excluding the source s and the sink t, for all time intervals starting at the starting timestamp and finishing before the ending timestamp, the incoming flow of F to u has a flow value not less than the flow value of the outgoing flow of F from u. Thus, from the starting timestamp to any timestamp, there is no node (excluding s and t) has more outgoing flow than its incoming flow.

For a temporal flow F from source s at starting timestamp τ_s to sink t at ending timestamp τ_e, F's *flow value* $|F|$ is,

$$|F|=\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]}F(s,v,\tau)=\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]}F(v,t,\tau)\qquad(5)$$

Moreover, the *flow density* (or simply *density*) of F is defined by,

$$|F|/(\tau_e-\tau_s)\qquad(6)$$

Table 1 lists some frequently used notations. In the following, we define a *novel* concept of a bursting flow in temporal flow networks called *δ-bursting flow* and then present the problem statement.

Table 1: Frequently used notations

Notation	Description
s; t	the source node; the sink node
δ; T	the bursting parameter for δ-BFlow; sequence of timestamps
N; N_T	flow network; temporal flow network
C; C_T	capacity mapping function of N; capacity mapping function of N_T
f; F	flow; temporal flow
$\lvert f\rvert$; $\lvert F\rvert$	flow value of flow f; flow value of temporal flow F
N_f	the residual network of N *w.r.t.* f
p; Flow(p)	augmenting path; the value of the flow passes through augmenting path p
τ_s; τ_e	the starting timestamp; the ending timestamp
$[\tau,\tau']$	time interval where $\tau,\tau'\in T$ and $\tau\leq\tau'$
$N_{[\tau,\tau']}$	the transformed flow network of N_T *w.r.t.* $[\tau,\tau']$
u; $\langle u,\tau\rangle$	node in N or N_T; node in transformed flow network
(u,v); (u,v,τ)	edge from u to v in N; edge from u to v at timestamp τ in N_T
$(\langle u,\tau\rangle,\langle v,\tau'\rangle)$	edge from $\langle u,\tau\rangle$ to $\langle v,\tau'\rangle$ in transformed flow network
$\text{TiStamp}_{\text{out}}(u)$	set of the timestamps of the out-going edges of u in N_T
$\text{TiStamp}_{\text{in}}(u)$	set of the timestamps of the in-coming edges of u in N_T
Ti(u)	set of the timestamps of the edges of u in N_T *that may be part of flows from s to t*: Ti(s) = $\text{TiStamp}_{\text{out}}(s)$, Ti($t$) = $\text{TiStamp}_{\text{in}}(t)$, and Ti($u$) = $\text{TiStamp}_{\text{out}}(u)\cup\text{TiStamp}_{\text{in}}(u)$, $u\neq s,t$
Seq(Ti)	sequence consisting of the elements in Ti in the ascending order
MF$[\tau_s,\tau_e]$	Maxflow from $\langle s,\tau_s\rangle$ to $\langle t,\tau_e\rangle$ in $N_{[\tau'_s,\tau'_e]}$ ($\tau_s\geq\tau'_s$ and $\tau_e\leq\tau'_e$)
$d_{\max}$	the maximum degree of nodes of a temporal flow network
d	max $\{\lvert\text{Ti}(s)\rvert,\lvert\text{Ti}(t)\rvert\}$
$\uplus$	binary operator to combine transformed flow networks
$\setminus$	binary operator to subtract one transformed flow network from another
$\Lambda_\tau(N)$	timestamp injection operator on transformed flow network N for τ

[3] We omit Dinic's details related to the *level graph* and *blocking flow*, and show only the augmenting paths, since our proposed solutions are built *on top* of the computation of augmenting paths. Interested readers can find more details in [12].

Definition 2. (δ-***bursting flow*** (δ-BFlow)) *Given a temporal flow network $N_T = (V, E_T, T, C_T)$, a source s, a sink t, a bursting parameter δ for bursting time interval, the* δ-bursting flow *is a temporal flow F from s at a starting timestamp τ_s to t at an ending timestamp τ_e that satisfies the conditions:*

a) $\tau_e - \tau_s \geq \delta$; and
b) the flow density of F is maximized.

For a δ-BFlow F, we denote $[\tau_s, \tau_e]$ as F's *bursting interval*. To simplify the pseudo code in this work, we define a binary record $\big(|F|/(\tau_e - \tau_s), [\tau_s, \tau_e]\big)$ of F, where $|F|/(\tau_e - \tau_s)$ and $[\tau_s, \tau_e]$ are F's density and bursting interval, respectively.

Example 3. *Figure 3(a) shows an example of a temporal flow network $N_T = (V, E_T, T, C_T)$, where $T = [1, 2, 3, 4, 5, 6]$ and the formatted data on each edge (u, v, τ) denotes $C_T(u, v, \tau)|\tau$; Figure 3(b) shows in (blue) dashed lines a temporal flow F from s at timestamp 3 to t at timestamp 6 in N_T, where $F(s, v_2, 3) = F(v_2, v_3, 4) = F(v_3, v_4, 4) = F(v_4, t, 4) = 2$, $F(s, v_2, 5) = F(v_2, v_3, 6) = F(v_3, v_5, 6) = F(v_5, t, 6) = 1$, and $F(u, v, \tau) = 0$, otherwise. The flow value $|F|$ of F is 3. We can also see an example of the time constraint on F that, for node v_2 and timestamp 5, $\sum_{v \in V} \sum_{\tau \in [1,5]} F(v, v_2, \tau) = 3 \geq \sum_{v \in V} \sum_{\tau \in [1,5]} F(v_2, v, \tau) = 2$; and Figure 3(c) shows in (red) dotted lines a 2-BFlow F' from s at 1 to t at 4 in N_T having $|F'| = 5$ and bursting interval $[1, 4]$. F' has a density of $5/3$, which is larger than the densities of any other temporal flows in N_T, e.g., the density $3/3$ of the temporal flow F shown in Figure 3(b). Therefore, F' is a 2-BFlow.*

Problem statement. *Given a temporal flow network $N_T = (V, E_T, T, C_T)$ and a bursting flow query $Q = (s, t, \delta)$ where $s, t \in V$, the goal is to find the density and bursting interval of the δ-BFlow from s to t in N_T.*

4 A Practical Solution for δ-BFlow

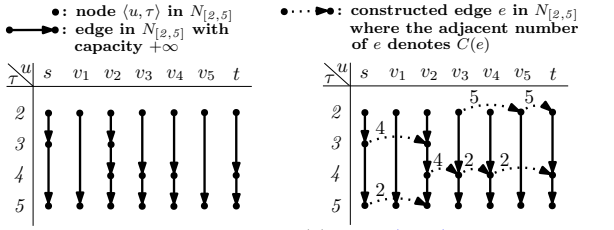
In this section, we propose the *first practical Bursting Flow Query Solution* called BFQ that consists of i) *the network transformation procedure* (Section 4.1) that converts the problem of finding δ-BFlow on temporal flow network N_T into finding Maxflows on transformed flow networks of N_T *w.r.t.* all the candidate time intervals; and ii) *the candidate interval enumeration procedure* (Section 4.2) that reduces the number of candidate time intervals in practice. Section 4.3 illustrates how BFQ computes δ-BFlows in a temporal flow network.

4.1 Network Transformation

In this subsection, we first introduce the *network transformation* procedure that transforms a temporal flow network N_T into a flow network *w.r.t.* a time interval. Then, we prove that finding δ-BFlow in N_T is *equivalent* to finding Maxflows in N_T's transformed flow networks *w.r.t.* all possible time intervals.

Consider a temporal flow network $N_T = (V, E_T, T, C_T)$, a source s at the starting timestamp τ_s and a sink t at the ending timestamp τ_e. We first introduce the *timestamp-inline* operator.

Timestamp inlining. For any node u in V, let $\mathrm{TiStamp}_{\mathrm{out}}(u) = \{\tau \mid (u, v, \tau) \in E\}$ denote the set of the timestamps of the outgoing edges of u in N_T (similarly, $\mathrm{TiStamp}_{\mathrm{in}}(u)$ denotes the case for incoming edges) and let Seq(Ti) denote a sequence consisting of the timestamps in set Ti in ascending order. Then, the *timestamp-inline operator* on u for time interval $[\tau_s, \tau_e]$ has the following steps:



(a) Steps 1) & 2): timestamp inlining on nodes in N_T for $[2, 5]$ (b) Steps 2) & 3): construction of edges and capacities for $N_{[2,5]}$

Figure 4: Example of transforming N_T in Figure 3(a) into the transformed flow network $N_{[2,5]}$ of N_T *w.r.t.* $[2, 5]$

1) initialize set $\mathrm{Ti}(u)$ of the timestamps of the edges in N_T that are incident on u and may be part of flows from s to t:
– $\mathrm{Ti}(s) = \mathrm{TiStamp}_{\mathrm{out}}(s)$, $\mathrm{Ti}(t) = \mathrm{TiStamp}_{\mathrm{in}}(t)$; and
– $\mathrm{Ti}(u) = \mathrm{TiStamp}_{\mathrm{out}}(u) \cup \mathrm{TiStamp}_{\mathrm{in}}(u)$, $u \in V - \{s, t\}$;
2) let $\mathrm{Ti}_{[\tau_s,\tau_e]}(u) = \big\{\tau \mid \tau \in \mathrm{Ti}(u) \cup \{\tau_s, \tau_e\} \text{ and } \tau_s \leq \tau \leq \tau_e\big\}$, denote the set of timestamps in $\mathrm{Ti}(u)$ within time interval $[\tau_s, \tau_e]$;
3) for each timestamp τ in $\mathrm{Ti}_{[\tau_s,\tau_e]}(u)$, create a node $\langle u, \tau\rangle$; and
4) for each timestamp τ and the succeeding timestamp τ' of τ in $\mathrm{Seq}\big(\mathrm{Ti}_{[\tau_s,\tau_e]}(u)\big)$, create an edge $\big(\langle u, \tau\rangle, \langle u, \tau'\rangle\big)$.

Network transformation procedure. For time interval $[\tau_s, \tau_e]$, the transformed flow network of N_T *w.r.t.* $[\tau_s, \tau_e]$, denoted by $N_{[\tau_s,\tau_e]} = (V', E, C)$, is a flow network, where V' and E are first initialized as empty sets. Then, the *network transformation* procedure constructs $N_{[\tau_s,\tau_e]}$ with the following steps.

1) **Timestamp inlining on the source and sink.** Firstly, we conduct timestamp inlining on the source s and sink t for $[\tau_s, \tau_e]$, and then insert the created nodes and edges into V' and E, respectively.
2) **Timestamp inlining on nodes and construction of edges.** Secondly, starting from s, we perform a depth-first traversal on the edges of N_T having timestamps within $[\tau_s, \tau_e]$. For each traversed edge (u, v, τ), i) if $\langle v, \tau\rangle \notin V'$, we conduct timestamp inlining on v for $[\tau_s, \tau_e]$, and insert the created nodes and edges into V' and E, respectively; and ii) we create and insert the edge $\big(\langle u, \tau\rangle, \langle v, \tau\rangle\big)$ into E.
3) **Construction of capacities.** Lastly, we construct the capacity mapping function C of $N_{[\tau_s,\tau_e]}$ as follows:
① for each edge e in $N_{[\tau_s,\tau_e]}$ constructed by the traversed edge e' in N_T in step 2), e has the same capacity as that of e'. That is,
$$\forall\, \langle u, \tau\rangle, \langle v, \tau\rangle \in V', C\big(\langle u, \tau\rangle, \langle v, \tau\rangle\big) = C_T(u, v, \tau);$$
② for the edges created by the timestamp-inline operator, their capacities are constructed as $+\infty$ since there can always be a flow from a timestamp to the later one on the same node. That is,
$$\forall\, \langle u, \tau\rangle, \langle u, \tau'\rangle \in V', \text{if } \big(\langle u, \tau\rangle, \langle u, \tau'\rangle\big) \in E, C\big(\langle u, \tau\rangle, \langle u, \tau'\rangle\big) = +\infty;$$
③ for the non-existing edges, their capacities are constructed as 0:
$$\forall\, \langle u, \tau\rangle, \langle v, \tau'\rangle \in V' \ (u \neq v \text{ and } \tau \neq \tau'), C\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) = 0; \text{ and}$$
$$\forall\, \langle u, \tau\rangle, \langle u, \tau'\rangle \in V', \text{if } \big(\langle u, \tau\rangle, \langle u, \tau'\rangle\big) \notin E, C\big(\langle u, \tau\rangle, \langle u, \tau'\rangle\big) = 0.$$

Example 4. *Figure 4 shows an example of transforming the temporal flow network N_T in Figure 3(a) into a transformed flow network w.r.t. $[2, 5]$. First, steps 1) and 2) of the network transformation procedure conduct timestamp inlining on source s, sink t, and the nodes of N_T that are reachable from s during $[2, 5]$. For N_T, as $\mathrm{Seq}(\mathrm{Ti}(s)) = [2, 3, 5]$, $\mathrm{Seq}\big(\mathrm{Ti}(v_1)\big) = \mathrm{Seq}\big(\mathrm{Ti}(v_5)\big) = [2, 5]$, $\mathrm{Seq}\big(\mathrm{Ti}(v_2)\big) = [2, 3, 4, 5]$, and $\mathrm{Seq}\big(\mathrm{Ti}(v_3)\big) = \mathrm{Seq}\big(\mathrm{Ti}(v_4)\big) = \mathrm{Seq}\big(\mathrm{Ti}(t)\big) = [2, 4, 5]$, the timestamp inlining on nodes constructs the nodes and edges as shown*

in Figure 4(a). Meanwhile, for the traversed edges of N_T during $[2, 5]$ in step 2), steps 2) and 3) construct edges in dotted lines as shown in Figure 4(b), where the adjacent number denotes the capacity.

Given a temporal flow network $N_T = (V, E_T, T, C_T)$, a source s at τ_s, and a sink t at τ_e, the network transformation procedure can transform N_T into a transformed flow network $N_{[\tau_s,\tau_e]}$ in $O\left(d_{\max} \cdot |V| + |E_T|\right)$ time, where $d_{\max}$ is the maximum degree of nodes of N_T since $N_{[\tau_s,\tau_e]}$ has $O(d_{\max} \cdot |V|)$ nodes and $O(d_{\max} \cdot |V| + |E_T|)$ edges. Then, we have the following lemma indicating that there exists a temporal flow in N_T and a flow in $N_{[\tau_s,\tau_e]}$ having the same flow value.

Lemma 1. (**Equivalent flow of the same flow value**) *Consider a temporal flow network N_T, a source s at the starting timestamp τ_s, a sink t at the ending timestamp τ_e, and the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T w.r.t. $[\tau_s, \tau_e]$. Given any temporal flow F from s to t in N_T, there exists a flow f from node $\langle s, \tau_s\rangle$ to node $\langle t, \tau_e\rangle$ in $N_{[\tau_s,\tau_e]}$ such that $|f| = |F|$, and vice versa.*

Lemma 1 can be proved by constructing an equivalent flow f in $N_{[\tau_s,\tau_e]}$ for F and an equivalent temporal F in N_T for f such that $|f| = |F|$. The detailed construction is presented in Appendix B [1].

According to Definition 2, the flow densities of temporal flows from source s at τ_s to sink t at τ_e in a temporal flow network N_T during the specific time interval $[\tau_s, \tau_e]$ are determined by their flow values. Therefore, finding the temporal flow in N_T during $[\tau_s, \tau_e]$ whose flow density is maximized is *equivalent* to finding the temporal flow F having the maximum flow value during $[\tau_s, \tau_e]$. With Lemma 1, F can be found by finding Maxflow in the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T *w.r.t.* $[\tau_s, \tau_e]$. In this way, δ-BFlow from s to t in N_T can be found by finding Maxflows in N_T's transformed flow networks *w.r.t.* all possible time intervals. Note that $N_{[\tau_s,\tau_e]}$ is a subgraph of the transformed flow network $N_{[\tau'_s,\tau'_e]}$ of N_T *w.r.t.* $[\tau'_s, \tau'_e]$, where $\tau_s \geq \tau'_s$ and $\tau_e \leq \tau'_e$. For simplicity, Maxflow from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e\rangle$ in $N_{[\tau'_s,\tau'_e]}$ ($\tau_s \geq \tau'_s$ and $\tau_e \leq \tau'_e$) is called Maxflow *w.r.t.* $[\tau_s, \tau_e]$, denoted by $\mathsf{MF}[\tau_s, \tau_e]$, when it is clear from the context.

4.2 Candidate Interval Enumeration

In this subsection, we present the procedure for enumerating all the candidate time intervals since δ-BFlow can be found by finding Maxflows in the transformed flow networks *w.r.t.* these intervals.

A naive method is to enumerate all the possible $O(|T|^2)$ time intervals $[\tau_s, \tau_e]$, $\tau_s, \tau_e \in T$ as *candidate intervals*, where T is the timestamp sequence of a temporal flow network. However, the size of T of a temporal network is usually large. For example, the dataset of the bitcoin transaction network in 2011 (Btc2011) has $59K$ timestamps that such enumeration is impractical. To reduce the size of the enumeration of candidate intervals, we propose to enumerate the *core intervals* for finding δ-BFlow.

Core interval. Given a temporal flow network $N_T = (V, E_T, T, C_T)$ with source s and sink t, a time interval $[\tau_s, \tau_e]$, $\tau_s, \tau_e \in T$ is a *core interval* if, in $N_{[\tau_s,\tau_e]}$, the flow value $\left|\mathsf{MF}[\tau_s, \tau_e]\right|$ of $\mathsf{MF}[\tau_s, \tau_e]$ is larger than the flow values of Maxflows *w.r.t.* any subintervals $[\tau'_s, \tau'_e]$ of $[\tau_s, \tau_e]$ (*i.e.*, $\tau'_s \geq \tau_s, \tau'_e < \tau_e$ or $\tau'_s > \tau_s, \tau'_e \leq \tau_e$).

For example, $[3, 4]$ is a core interval in the temporal flow network shown in Figure 3(a) since $\left|\mathsf{MF}[3, 4]\right| = 2 > \left|\mathsf{MF}[3, 3]\right| = \left|\mathsf{MF}[4, 4]\right| = 0$. Then, we have the following observation for core intervals.

Algorithm 1: δ-BFlow Query Solution (BFQ)

Input : A temporal flow network $N_T = (V, E_T, T, C_T)$, a source s, a sink t, and a bursting parameter δ
Output: The binary record of a δ-BFlow from s to t in N_T

1 $r \leftarrow 0$; //r: flow density of current δ-BFlow
2 $\tau_{r_s} \leftarrow 0, \tau_{r_e} \leftarrow 0$; //$[\tau_{r_s}, \tau_{r_e}]$: bursting interval of current δ-BFlow
3 **foreach** $\tau_s \in \mathsf{Ti}(s)$ **do**
4 **foreach** $\tau_e \in \mathsf{Ti}(t) \cup \{\tau_s + \delta\}$ **do** // corner case may exist[4]
5 $N_{[\tau_s,\tau_e]} \leftarrow \mathsf{NetworkTrans}(N_T, s, t, \tau_s, \tau_e)$; //Section 4.1
6 invoke Dinic on $N_{[\tau_s,\tau_e]}$ to compute a Maxflow f *w.r.t.* $[\tau_s, \tau_e]$;
7 **if** $|f|/(\tau_e - \tau_s) > r$ **then**
8 $r \leftarrow |f|/(\tau_e - \tau_s), \tau_{r_s} \leftarrow \tau_s, \tau_{r_e} \leftarrow \tau_e$; //update of δ-BFlow
9 **return** $(r, [\tau_{r_s}, \tau_{r_e}])$; //output binary record of δ-BFlow

Observation 1. (**Core interval existence**) *Given a temporal flow network N_T with source s and sink t, if a time interval $[\tau_s, \tau_e]$ is a core interval, then $\tau_s \in \mathsf{TiStamp}_{\mathsf{out}}(s)$ and $\tau_e \in \mathsf{TiStamp}_{\mathsf{in}}(t)$.*

Observation 1 can be proved as follows. If τ_s is not the timestamp of an out-going edge of s in N_T (*i.e.*, $\tau_s \notin \mathsf{TiStamp}_{\mathsf{out}}(s)$), there must exist a time interval $[\tau'_s, \tau_e]$, $\tau'_s > \tau_s$ such that $\left|\mathsf{MF}[\tau'_s, \tau_e]\right| = \left|\mathsf{MF}[\tau_s, \tau_e]\right|$. Similar time interval exists if $\tau_e \notin \mathsf{TiStamp}_{\mathsf{in}}(t)$.

With Observation 1, we can know that there exist $O(d^2)$ core intervals for a δ-BFlow query $Q = (s, t, \delta)$ in a temporal flow network $N_T = (V, E_T, T, C_T)$, where $d = \max\left\{|\mathsf{Ti}(s)|, |\mathsf{Ti}(t)|\right\}$. Based on the core interval, we present the following lemma to prune the enumeration of all the $O(|T|^2)$ possible time intervals.

Lemma 2. *Given a δ-BFlow query $Q = (s, t, \delta)$ on a temporal flow network N_T, δ-BFlow in N_T for Q can be found by finding $\mathsf{MF}[\tau_s, \tau_e]$, where τ_s and τ_e satisfy either*

- *$\tau_e = \tau_s + \delta$, $\tau_s \in \mathsf{Ti}(s)$; or*
- *$[\tau_s, \tau_e]$ is a core interval such that $\tau_e - \tau_s > \delta$.*

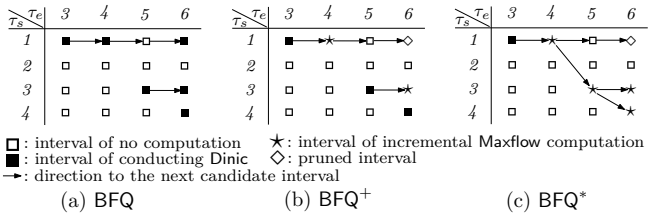
Lemma 2 can be proved by showing that δ-BFlows have the same flow density while the bursting intervals of the found δ-BFlows can be used to obtain the bursting intervals of the rest δ-BFlows in a simple method. Based on Lemma 2, we can enumerate at most $O(d^2)$ candidate intervals to find δ-BFlow. As shown in Table 2, d of the three evaluated real world datasets are often smaller than 100.

4.3 Practical δ-BFlow Solution (BFQ)

Putting these together, we present a practical δ-BFlow solution called BFQ. BFQ contains the two above mentioned procedures:

- **Candidate interval enumeration procedure (Section 4.2).** Given a δ-BFlow query $Q = (s, t, \delta)$ on a temporal flow network, we enumerate $O(d^2)$ core intervals and $|\mathsf{Ti}(s)|$ time intervals $[\tau, \tau + \delta]$, $\tau \in \mathsf{Ti}(s)$ as the candidate intervals, where d equals to $\max\left\{|\mathsf{Ti}(s)|, |\mathsf{Ti}(t)|\right\}$.
- **Network transformation procedure (Section 4.1).** We transform the temporal flow network as the transformed flow network *w.r.t.* each candidate interval. δ-BFlow can be found by computing Maxflows in these transformed flow networks.

In detail, Algorithm 1 shows BFQ for finding the flow density and time interval of δ-BFlow. Taking as inputs a temporal flow network $N_T = (V, E_T, T, C_T)$, a source s, a sink t, and the bursting parameter δ, Algorithm 1 outputs the flow density and the bursting interval of a δ-BFlow from s to t in N_T. Note that all the bursting intervals can be obtained with minor modifications. First, Lines 1-2 initialize the current maximum flow density r and its corresponding bursting interval $[\tau_{r_s}, \tau_{r_e}]$ as 0 and $[0, 0]$, respectively. Lines 3-4 enumerate



(a) BFQ (b) BFQ$^+$ (c) BFQ*

Figure 5: Patterns of Maxflow computation of proposed algorithms on the space of candidate intervals $[\tau_s, \tau_e]$ for computing 2-BFlow from s to t in the N_T shown in Figure 3(a)

candidate intervals with simple corner cases[4] according to the candidate interval enumeration procedure (Section 4.2). For each candidate interval $[\tau_s, \tau_e]$, Line 5 transforms N_T into a transformed flow network $N_{[\tau_s,\tau_e]}$ *w.r.t.* $[\tau_s, \tau_e]$ (Section 4.1). Line 6 computes an $\mathsf{MF}[\tau_s, \tau_e]$ (denoted by f) from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e\rangle$ in $N_{[\tau_s,\tau_e]}$ by using the practical solution Dinic [12]. f's flow density is computed in Line 7. Line 8 updates r and $[\tau_{r_s}, \tau_{r_e}]$ by using the current maximum flow density and its corresponding time interval, respectively. Line 9 returns r and $[\tau_{r_s}, \tau_{r_e}]$ as outputs.

Example 5. *For the temporal flow network N_T in Figure 3(a),* BFQ *computes* 2-BFlow *from s to t in N_T by conducting* Dinic *on the transformed flow networks w.r.t. candidate intervals $[\tau_s, \tau_e]$ as shown in Figure 5(a). Note that the evaluation on candidate interval $[4, 6]$ is a corner case (Line 4 of Algorithm 1) since $5 \in \mathsf{Ti}(s)$ and $5 + \delta = 7 > 6$.* BFQ *can find one* 2-BFlow *from s to t having the flow density $5/3$ and the bursting interval $[1, 4]$ as shown in Figure 3(c).*

Analysis. With Lemma 1, BFQ correctly computes the flow density and the bursting interval of δ-BFlow in a temporal flow network $N_T = (V, E_T, T, C_T)$ by computing Maxflows *w.r.t.* all the $O(d^2)$ candidate intervals. The time complexity of BFQ is $O\Big(d^2 \cdot \big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$. In the next section, we propose optimizations to avoid some redundant computation in BFQ.

5 Efficient δ-BFlow Solution — Incremental Augmenting Path Approach

In Section 4, we propose a δ-BFlow solution BFQ that enumerates $O(d^2)$ candidate intervals (Lines 3-4 of Algorithm 1) for Maxflow evaluations. According to the network transformation procedure of BFQ (Section 4.1), it is apparent that, for any two enumerated intervals $[\tau'_s, \tau'_e]$ and $[\tau_s, \tau_e]$, if $[\tau'_s, \tau'_e]$ is a subinterval of $[\tau_s, \tau_e]$, the transformed flow network $N_{[\tau'_s,\tau'_e]}$ is a subgraph of the transformed flow network $N_{[\tau_s,\tau_e]}$. Therefore, the augmenting paths found in $N_{[\tau'_s,\tau'_e]}$ for $\mathsf{MF}[\tau'_s, \tau'_e]$ also exist in $N_{[\tau_s,\tau_e]}$ as augmenting paths for $\mathsf{MF}[\tau_s, \tau_e]$, which leads an opportunity for incremental Maxflow computation. In this section, we propose incremental computation strategies for two distinguished cases, namely insertion case (section 5.1) and deletion case (section 5.2), as their details are different.

[4]Note that there may exist a corner case when the value of τ_e in Line 4 is larger than the maximum timestamp in T. The candidate interval for this case can be replaced by $[\tau_e - \delta, \tau_e]$ according to the analysis in Section 4.2. For simplicity, we omit this corner case in the following sections since it does not affect the analysis and correctness.

5.1 BFQ with Incremental Maxflow Computation of the Insertion Case (BFQ$^+$)

In this subsection, we propose an optimization to incrementally compute Maxflows *w.r.t.* the candidate intervals $[\tau_s, \tau_e]$ when increasing the value of τ_e with a fixed τ_s (Lines 4 of Algorithm 1).

5.1.1 *Operators for Incremental* Maxflow *computation.*

According to the network transformation procedure (Section 4.1), the increase of τ_e leads to only insertion of nodes and edges in the corresponding transformed flow network without the deletion of nodes and edges, and hence, the augmenting paths found for Maxflow *w.r.t.* a given $[\tau_s, \tau_e]$ are also valid augmenting paths for $\mathsf{MF}[\tau_s, \tau'_e]$ $(\tau'_e > \tau_e)$. For better presentation of the relation between the transformed flow networks *w.r.t.* time intervals of a fixed τ_s and different τ_es, we first define two binary operators on transformed flow networks.

Operator $\uplus$. Given two transformed flow networks $N_a = (V_a, E_a, C_a)$ and $N_b = (V_b, E_b, C_b)$, $N_a \uplus N_b$ denotes a flow network $N = (V_a \cup V_b, E_a \cup E_b, C)$, where $C\colon (V_a \cup V_b) \times (V_a \cup V_b) \to \mathbb{R}_{\geq 0}$ satisfies that $\forall\, \langle u, \tau\rangle, \langle v, \tau'\rangle \in V_a \cup V_b$, $C\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big)$ equals

$$\begin{cases} C_a\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) + C_b\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big), \text{ if } \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E_a \cap E_b; \\ C_a\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big), \text{ if } \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E_a - E_b; \\ C_b\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big), \text{ if } \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E_b - E_a; \\ 0, \text{ otherwise.} \end{cases}$$

Operator $\setminus$. Given two transformed flow networks $N_a = (V_a, E_a, C_a)$ and $N_b = (V_b, E_b, C_b)$, assume that N_a and N_b are the left operand and the right operand of the operator $\setminus$, respectively. Then, $N_a \setminus N_b$ denotes a flow network $N = (V, E, C)$, where i) $E = (E_a - E_b) \cup (E_a \cap E_b)$, ii) $V = \Big\{\langle u, \tau\rangle \mid \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E \text{ or } \big(\langle v, \tau'\rangle, \langle u, \tau\rangle\big) \in E\Big\}$, and iii) $C\colon V \times V \to \mathbb{R}_{\geq 0}$ satisfies that $\forall\, \langle u, \tau\rangle, \langle v, \tau'\rangle \in V$, $C\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big)$ equals

$$\begin{cases} C_a\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) - C_b\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big), \text{ if } \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E_a \cap E_b; \\ C_a\big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big), \text{ if } \big(\langle u, \tau\rangle, \langle v, \tau'\rangle\big) \in E_a - E_b; \\ 0, \text{ otherwise.} \end{cases}$$

Operator $\uplus$ generates a union of two flow networks by merging the capacities of their common edges, while operator $\setminus$ generates a flow network of the left operand by reducing the capacities of common edges based on the capacities of the edges of the right operand. (Examples of these two operators will be given together with the examples of the incremental Maxflow). Note that both operators take $O(|E|)$ times only. Then, we propose the following lemma to incrementally compute $\mathsf{MF}[\tau_s, \tau_e]$ *when increasing the value of τ_e*. We call it the insertion case.

5.1.2 *Incremental* Maxflow *computation (insertion case).*

Lemma 3. (**Incremental** Maxflow (**insertion case**)) *Consider a temporal flow network N_T with source s and sink t, the transformed flow network $N_{[\tau,\tau_e]}$ of N_T w.r.t. $[\tau, \tau_e]$, and the following notations.*

- *$f_{[\tau_s,\tau_e]}$ denotes an $\mathsf{MF}[\tau_s, \tau_e]$ (where $\tau_s \geq \tau$) in $N_{[\tau,\tau_e]}$, obtained by* Dinic *by finding augmenting paths; and*
- *τ'_e is a timestamp larger than τ_e.*

With these notations, a Maxflow *$f_{[\tau_s,\tau'_e]}$ w.r.t. $[\tau_s, \tau'_e]$ in $N_{[\tau,\tau'_e]}$ can be obtained by finding augmenting paths $p_{\tau'_e}$s from $\langle s, \tau_s\rangle$ to*

Algorithm 2: The Improved δ-BFlow Query Solution (BFQ$^+$)

Input : A temporal flow network $N_T = (V, E_T, T, C_T)$, a source s, a sink t, and a bursting parameter δ
Output: The binary record of a δ-BFlow from s to t in N_T

1 $r \leftarrow 0$; //r: flow density of current δ-BFlow
2 $\tau_{r_s} \leftarrow 0, \tau_{r_e} \leftarrow 0$; //$[\tau_{r_s}, \tau_{r_e}]$: bursting interval of current δ-BFlow
3 **foreach** $\tau_s \in$ Ti(s) **do**
4 $\quad \tau_e \leftarrow \tau_s + \delta$;
5 $\quad N_{[\tau_s,\tau_e]} \leftarrow$ NetworkTrans$(N_T, s, t, \tau_s, \tau_e)$; //Section 4.1
6 $\quad$ invoke Dinic on $N_{[\tau_s,\tau_e]}$ to compute a Maxflow f *w.r.t.* $[\tau_s, \tau_e]$ and the residual network N_f *w.r.t.* f;
7 $\quad$ **if** $|f|/(\tau_e - \tau_s) > r$ **then**
8 $\quad\quad r \leftarrow |f|/(\tau_e - \tau_s), \tau_{r_s} \leftarrow \tau_s, \tau_{r_e} \leftarrow \tau_e$;//update of δ-BFlow
9 $\quad$ **foreach** τ'_e in Ti(t) that $\tau'_e > \tau_s + \delta$ **do** // corner case may exist[4]
10 $\quad\quad$ IncreMaxFlow$^+(N_T, f, N_f, \tau_s, \tau_e, \tau'_e, \tau_{r_s}, \tau_{r_e}, r)$; //Section 5.1
11 $\quad\quad \tau_e \leftarrow \tau'_e$;
12 **return** $(r, [\tau_{r_s}, \tau_{r_e}])$; //output binary record of δ-BFlow

Procedure IncreMaxFlow$^+(N_T, \&f, \&N_f, \tau_s, \tau_e, \tau'_e, \&\tau_{r_s}, \&\tau_{r_e}, \&r)$:
13 $N_{[\tau_e,\tau'_e]} \leftarrow$ NetworkTrans$(N_T, s, t, \tau_e, \tau'_e)$;//Section 4.1
14 $N_f \leftarrow N_f \uplus (N_{[\tau_e,\tau'_e]} \setminus N_{[\tau_e,\tau_e]})$; //Lemma 3:update N_f
15 **if** $|f| + \sum_{\tau\in[\tau_e,\tau'_e]} \sum_{u\in V} C_T(u,t,\tau) < r\cdot(\tau'_e - \tau_s)$ **then** // Observation 2
16 $\quad$ **return**;
17 invoke Dinic on N_f to update f as an MF$[\tau_s, \tau'_e]$ in $N_{[\tau_s,\tau'_e]}$; //Lemma 3
18 **if** $|f|/(\tau'_e - \tau_s) > r$ **then**
19 $\quad r \leftarrow |f|/(\tau'_e - \tau_s), \tau_{r_s} \leftarrow \tau_s, \tau_{r_e} \leftarrow \tau'_e$; //update of δ-BFlow

$\langle t, \tau'_e\rangle$ on the flow network $N_{f_{[\tau_s,\tau_e]}} \uplus (N_{[\tau_e,\tau'_e]} \setminus N_{[\tau_e,\tau_e]})$ having

$$|f_{[\tau_s,\tau'_e]}| = |f_{[\tau_s,\tau_e]}| + \sum_{p_{\tau'_e} s} \text{Flow}(p_{\tau'_e}),$$

where $N_{f_{[\tau_s,\tau_e]}}$ is the residual network of $N_{[\tau,\tau_e]}$ w.r.t. $f_{[\tau_s,\tau_e]}$.

Intuitively, Lemma 3 states that the augmenting paths for finding MF$[\tau_s, \tau_e]$ can be reused for finding MF$[\tau_s, \tau'_e]$ because the transformed flow network *w.r.t.* $[\tau_e, \tau'_e]$ are inserted into $N_{[\tau,\tau_e]}$ where the inserted edges have no effect on the reused augmenting paths. Therefore, we can incrementally compute MF$[\tau_s, \tau'_e]$, $\tau'_e \in \{\tau_s + \delta\} \cup$ Ti(t) without computing the common augmenting paths again. The detailed proof of Lemma 3 is presented in Appendix B [1]. Moreover, we propose a *capacity constraint* for pruning the Maxflow computation by the following observation.

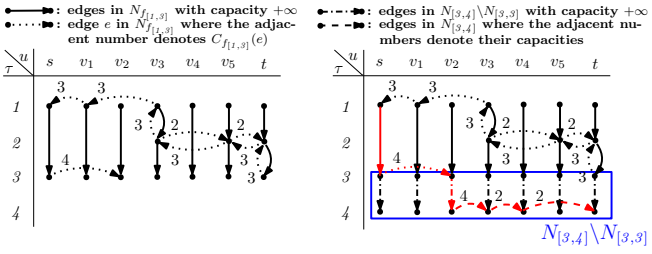
Observation 2. (**Capacity constraint**) *Given a temporal flow network $N_T = (V, E_T, T, C_T)$, and a source s and a sink t in N_T, let f denote an* MF$[\tau_s, \tau_e]$ *in the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T and r denote f's flow density $|f|/(\tau_e - \tau_s)$. For a timestamp τ'_e ($\tau'_e > \tau_e$), if* MF$[\tau_s, \tau'_e]$ *is a δ-*BFlow *from s to t, the inequality holds*

$$|f| + \sum_{\tau\in[\tau_e,\tau'_e]} \sum_{u\in V} C_T(u,t,\tau) \geq r \cdot (\tau'_e - \tau_s).$$

Observation 2 can be easily proved by contradiction. Assume that all the incoming edges of sink t in N_T having timestamps within $[\tau_e, \tau'_e]$ spend all their capacities for MF$[\tau_s, \tau'_e]$. If such a Maxflow corresponds to a temporal flow having a flow density smaller than r, it cannot be a δ-BFlow.

*5.1.3 Improved δ-*BFlow *solution by using the incremental* Maxflow *computation of the insertion case.* With Lemma 3 and Observation 2, we propose an improved δ-BFlow solution called BFQ$^+$. We put a plus sign to BFQ because we compute MF$[\tau_s, \tau_e]$ *once* for a fixed τ_s, and then increase the value of τ_e for the next iteration of the Maxflow computation as shown in Line 4 of Algorithm 1.

As shown in Algorithm 2, BFQ$^+$ has the same input and output as those of BFQ. After the initialization of the current δ-BFlow in Lines 1-2, Line 3 enumerates the starting timestamp τ_s for candidate



(a) The residual network $N_{f_{[1,3]}}$ of $N_{[1,3]}$ *w.r.t* an MF$[1, 3]$ denoted by $f_{[1,3]}$

(b) The flow network $N_{f_{[1,3]}} \uplus (N_{[3,4]}\setminus N_{[3,3]})$, where $N_{[3,4]}\setminus N_{[3,3]}$ is shown by the (blue) rectangle

Figure 6: [Best viewed in color] Example of the incremental Maxflow **computation (insertion case) in** BFQ$^+$ **from an** MF$[1, 3]$ **to an** MF$[1, 4]$

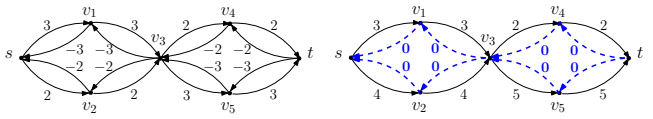
intervals $[\tau_s, \tau_e]$ according to the candidate interval enumeration procedure (Section 4.2). Importantly, for each enumerated τ_s, Lines 4-6 first initialize τ_e as $\tau_s + \delta$ and invoke Dinic once on the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T to compute a Maxflow f *w.r.t.* $[\tau_s, \tau_e]$. Lines 7-8 record the density r and the time interval $[\tau_{r_s}, \tau_{r_e}]$ of current δ-BFlow. Then, Lines 9-10 incrementally and iteratively compute MF$[\tau_s, \tau'_e]$, $\tau'_e \in$ Ti(t) by updating f as follows: i) Line 13 transforms N_T into $N_{[\tau_e,\tau'_e]}$; ii) Line 14 computes the flow network $N_f \uplus (N_{[\tau_e,\tau'_e]} \setminus N_{[\tau_e,\tau_e]})$ to update the current residual network N_f; iii) Lines 15-16 conduct a fast pruning on the Maxflow computation by using the capacity constraint in Observation 2; iv) Line 17 invokes Dinic on the updated N_f to update f as an MF$[\tau_s, \tau'_e]$; and v) Lines 18-19 update r and $[\tau_{r_s}, \tau_{r_e}]$ for recording the current δ-BFlow. After the incremental Maxflow computation in Line 10 finishes, Line 11 updates the value of τ_e by using the value of τ'_e for the next iteration of incremental Maxflow computation for the fixed τ_s. After evaluating Maxflows *w.r.t.* all the candidate intervals, r and $[\tau_{r_s}, \tau_{r_e}]$ are returned as outputs in Line 12.

Example 6. *Consider the computation of* BFQ$^+$ *for* 2-BFlow *from s to t in Figure 3(a)'s N_T. Figure 5(b) shows the pattern of candidate intervals evaluated by* BFQ$^+$. *According to Lemma 3,* BFQ$^+$ *computes* MF$[1, 4]$ *in an incremental method (Lines 13-19) as follows. Based on an* MF$[1, 3]$ *(denoted by $f_{[1,3]}$) and the residual network $N_{f_{[1,3]}}$ w.r.t. $f_{[1,3]}$,* BFQ$^+$ *computes the flow network $N_{f_{[1,3]}} \uplus (N_{[3,4]} \setminus N_{[3,3]})$. Figure 6(a) shows the residual network of $N_{[1,3]}$ w.r.t. $f_{[1,3]}$ while Figure 6(b) shows the flow network $N_{f_{[1,3]}} \uplus (N_{[3,4]} \setminus N_{[3,3]})$, where there exists an augmenting path p marked in red: $\langle s, 1\rangle \rightarrow \langle s, 3\rangle \rightarrow \langle v_2, 3\rangle \rightarrow \langle v_2, 4\rangle \rightarrow \langle v_3, 4\rangle \rightarrow \langle v_4, 4\rangle \rightarrow \langle t, 4\rangle$. Therefore, Line 17 invokes* Dinic *to update $f_{[1,3]}$ to obtain an* MF$[1, 4]$ *whose flow value is $|f_{[1,3]}| + \text{Flow}(p) = 3 + 2 = 5$. An* MF$[3, 6]$ *can be also computed in such an incremental method. Regarding the* Maxflow *computation for* MF$[1, 6]$, *it can be pruned in Lines 15-16 according to Observation 2. This is because an* MF$[1, 4]$ *(denoted by f) has flow value $|f| = 5$ and density $r = 5/3$, and hence, we have $|f| + \sum_{\tau\in[4,6]} \sum_{u\in V} C_T(u,t,\tau) = 5 + 1 < 5/3 \cdot (6 - 1)$.*

The following theorem states the correctness of BFQ$^+$.

Theorem 1. (**Correctness of** BFQ$^+$) *Given a temporal flow network N_T with source s and sink t, and a parameter δ,* BFQ$^+$ *returns the flow density and the bursting intervals of δ-*BFlow *from s to t.*

Theorem 1 can be proved by showing that BFQ$^+$ considers the Maxflow computation for the same candidate intervals enumerated in BFQ. The detailed proof is presented in Appendix B [1].



(a) The augmenting flow network $N(\{p\}) = (V', E', C')$, where p is the augmenting paths for the (blue) dashed flow in Figure 2(b) and the edge denotes C':$V' \times V' \to \mathbb{R}$

(b) The flow network $N \uplus N_f$, where N is the augmenting flow network shown in Figure 7(a), and N_f is the residual network shown in Figure 2(c)

Figure 7: An example of augmenting flow network

Analysis. For a temporal flow network $N_T = (V, E_T, T, C_T)$, Lines 5-8 take $O\Big(d \cdot \big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$ time for each τ_s. The update of residual network N_f in Lines 13-14 takes $O\Big(d^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$ time in total. The update of flow f by conducting Dinic on the updated N_f in Line 17 takes $O\Big(d^2 \cdot \big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$ time in total. Therefore, the time complexity of BFQ^+ is $O\Big(d^2 \cdot \big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$.

5.2 BFQ^+ with Incremental Maxflow Computation of the Deletion Case (BFQ^*)

In this subsection, we further propose an optimization to incrementally compute $\mathsf{MF}[\tau_s, \tau_s + \delta]$ when increasing the value of τ_s (Lines 3-6 of Algorithm 2). Given an $\mathsf{MF}[\tau_s, \tau_e]$ in $N_{[\tau_s, \tau_e]}$, different from the incremental Maxflow computation introduced in Section 5.1, the case of increasing the starting timestamp from τ_s to τ_s' ($\tau_s' > \tau_s$) may lead to the deletion of nodes and edges in the transformed flow network that such deletion will make some previously found augmenting paths being no longer reusable (*i.e.*, invalid) for $\mathsf{MF}[\tau_s', \tau_e]$.

We handle this case by *withdrawing the flows on such augmenting paths from* $\mathsf{MF}[\tau_s, \tau_e]$. We first define a new kind of flow network and an important operator on $N_{[\tau_s, \tau_e]}$ for τ_s'. By using this network and this operator, we can withdraw the flows on these invalid augmenting paths starting before τ_s', and hence, achieve an incremental Maxflow computation for $\mathsf{MF}[\tau_s', \tau_e]$ when increasing the value of the starting timestamp from τ_s to τ_s' ($\tau_s' > \tau_s$).

5.2.1 *Augmenting flow network and timestamp injection operator.*

Definition 3. (***Augmenting flow network***) *Consider a set P of augmenting paths in a flow network $N = (V, E, C)$. The* augmenting flow network *of P, denoted by $N(P) = (V', E', C')$, is a flow network where i) $E' = \big\{(u, v) \mid (u, v)$ or (v, u) is an edge in E passed through by $p, p \in P\big\}$, ii) V' consists of the nodes incident to the edges in E', and iii) C': $V' \times V' \to \mathbb{R}$ satisfies the following:*

- $\forall\, u, v \in V'$, *if* $(u, v) \notin E'$, $C'(u, v) = 0$; *otherwise*
- $C'(u, v) = \sum_{p \in \{p' \mid p' \in P \text{ and } p' \text{ passes through } (u, v)\}} \mathsf{Flow}(p) - \sum_{p \in \{p' \mid p' \in P \text{ and } p' \text{ passes through } (v, u)\}} \mathsf{Flow}(p)$.

Example 7. *Figure 7 shows an example of an augmenting flow network. For the flow f as shown in Figure 2(b), f can be considered as two augmenting paths p_1: $s \to v_2 \to v_3 \to v_4 \to t$ having* $\mathsf{Flow}(p_1) = 2$ *and p_2: $s \to v_1 \to v_3 \to v_5 \to t$ having* $\mathsf{Flow}(p_2) = 3$. *Figure 7(a) shows the augmenting flow network $N(P)$ of $P = \{p_1, p_2\}$ while Figure 7(b) shows the flow network $N(P) \uplus N_f$, where N_f is the residual network as shown in Figure 2(c). We can see that $N(P) \uplus N_f$ is the same as the flow network N in Figure 2(a) except for some additional edges in (blue) dashed lines having capacities equal to* 0.

As Example 7 shows, the augmenting flow network $N(P)$ can be used to withdraw the flows of the augmenting paths in P by conducting operator $\uplus$ between $N(P)$ and the residual network. Note that the additional edges in (blue) dashed lines in Figure 7(b) have capacities equal to 0, and hence, they have no effect on the augmenting path based Maxflow solutions. Then, we further define the timestamp injection operator on transformed flow networks to help enable the incremental Maxflow computation by withdrawing the flow values of the outdated augmenting paths when increasing the value of τ_s.

Timestamp injection. Given the transformed flow network $N_{[\tau_s, \tau_e]} = (V, E, C)$ *w.r.t.* $[\tau_s, \tau_e]$ and a timestamp τ_s' that $\tau_s < \tau_s' < \tau_e$, the *timestamp injection* on $N_{[\tau_s, \tau_e]}$ for τ_s', denoted by $\Delta_{\tau_s'}(N_{[\tau_s, \tau_e]})$, is to, for any u in $\big\{v \mid \langle v, \tau\rangle \in V$ and $\langle v, \tau_s'\rangle \notin V\big\}$, replace the edge $e = \big(\langle u, \tau_1\rangle, \langle u, \tau_2\rangle\big)$ having $\tau_1 < \tau_s' < \tau_2$ or $\tau_2 < \tau_s' < \tau_1$ by two edges $\big(\langle u, \tau_1\rangle, \langle u, \tau_s'\rangle\big)$ and $\big(\langle u, \tau_s'\rangle, \langle u, \tau_2\rangle\big)$, which have i) the same capacities as that of edge e; and ii) flows with the same flow value as that of the flow going through edge e.

Figure 8(a) shows an example of $\Delta_3(N_{f_{[1,4]}})$, where $N_{f_{[1,4]}}$ is the residual network *w.r.t.* Maxflow $f_{[1,4]}$ *w.r.t.* $[1, 4]$ in the transformed flow network $N_{[1,4]}$. The edges in (blue) dashed lines show the replaced edges after conducting the timestamp injection on $N_{f_{[1,4]}}$ for timestamp *3*. It can be easily derived from the definition of timestamp injection that conducting timestamp injection on a flow network N does not change to the flow values of augmenting paths in N.

With the augmenting flow network and timestamp injection operator, we propose the following lemma to incrementally compute $\mathsf{MF}[\tau_s, \tau_e]$ when increasing τ_s's value. As the flow on the invalid augmenting paths is to be withdrawed, we call it the deletion case.

5.2.2 *Incremental* Maxflow *computation (deletion case).*

Lemma 4. (**Incremental** Maxflow (**deletion case**)) *Consider a temporal flow network N_T with source s and sink t, the transformed flow network $N_{[\tau_s, \tau_e]}$ of N_T w.r.t. $[\tau_s, \tau_e]$, a timestamp $\tau_s' \in \mathsf{Ti}(s)$ that $\tau_s < \tau_s' < \tau_e$, and the following notations.*

- *$f_{[\tau_s, \tau_e]}$ denotes an $\mathsf{MF}[\tau_s, \tau_e]$ in $N_{[\tau_s, \tau_e]}$ obtained by* Dinic *by finding augmenting paths; and*
- *P denotes the set of the found augmenting paths that do not pass through node $\langle s, \tau_s'\rangle$.*

With these notations, a Maxflow *$f_{[\tau_s', \tau_e]}$ w.r.t. $[\tau_s', \tau_e]$ in $N_{[\tau_s', \tau_e]}$ can be obtained by finding augmenting paths $p_{\tau_s'}$s from $\langle s, \tau_s'\rangle$ to $\langle t, \tau_e\rangle$ in $\Big(\Delta_{\tau_s'}\big(N_{f_{[\tau_s, \tau_e]}} \uplus N(P)\big)\Big) \setminus \big(N_{[\tau_s, \tau_s']} \setminus N_{[\tau_s', \tau_s']}\big)$ having,*

$$|f_{[\tau_s', \tau_e]}| = |f_{[\tau_s, \tau_e]}| + \sum_{p_{\tau_s'}s} \mathsf{Flow}(p_{\tau_s'}) - \sum_{p \in P} \mathsf{Flow}(p)\,,$$

where $N_{f_{[\tau_s, \tau_e]}}$ is the residual network of $N_{[\tau_s, \tau_e]}$ w.r.t. $f_{[\tau_s, \tau_e]}$.

Intuitively, there exist augmenting paths that are no longer valid after increasing the starting timestamp from τ_s to τ_s' that the transformed flow network *w.r.t.* $[\tau_s, \tau_s']$ is removed from $N_{[\tau_s, \tau_e]}$. Lemma 4 states that we can withdraw the flow of these paths from the residual network of $N_{[\tau_s, \tau_e]}$ as the flow network $N_{f_{[\tau_s, \tau_e]}} \uplus N(P)$ and then just compute augmenting paths in the flow network $\Big(\Delta_{\tau_s'}\big(N_{f_{[\tau_s, \tau_e]}} \uplus N(P)\big)\Big) \setminus \big(N_{[\tau_s, \tau_s']} \setminus N_{[\tau_s', \tau_s']}\big)$ to find an $\mathsf{MF}[\tau_s', \tau_e]$ without computing all the augmenting paths from scratch. The detailed proof of Lemma 4 is presented in Appendix B [1].

Algorithm 3: The Optimized δ-BFlow Query Solution (BFQ*)

Input : A temporal flow network $N_T = (V, E_T, T, C_T)$, a source s, a sink t, and a bursting parameter δ
Output: The binary record of a δ-BFlow from s to t in N_T

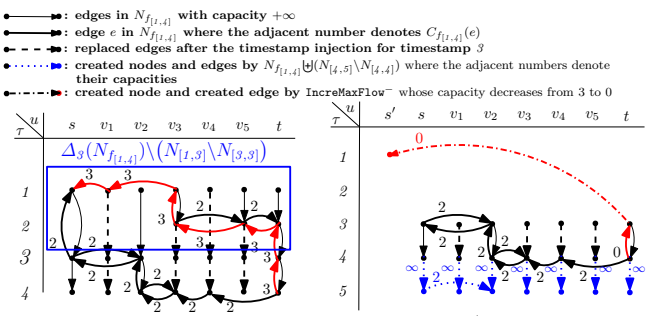
1 $r \leftarrow 0$; //r: flow density of current δ-BFlow
2 $\tau_{r_s} \leftarrow 0, \tau_{r_e} \leftarrow 0$; //$[\tau_{r_s}, \tau_{r_e}]$: bursting interval of current δ-BFlow
3 $\tau_s \leftarrow \min_{\tau \in \mathrm{Ti}(s)} \{\tau\}, \tau_e \leftarrow \min_{\tau \in \mathrm{Ti}(s)} \{\tau\} + \delta$;
4 $N_{[\tau_s,\tau_e]} \leftarrow \mathtt{NetworkTrans}(N_T, s, t, \tau_s, \tau_e), N_f \leftarrow N_{[\tau_s,\tau_e]}$; //Section 4.1
5 `UpdateMaxFlow`$(N_f, f, \tau_s, \tau_e, \tau_{r_s}, \tau_{r_e}, r)$;
6 **foreach** $\tau'_s \in \mathrm{Ti}(s)$ **do**
7 $flag \leftarrow$ **true**;
8 **foreach** τ'_e in $\mathrm{Ti}(t)$ that $\tau'_e > \tau_s + \delta$ **do** // corner case may exist[4]
9 **if** $(\tau'_e > \tau'_s + \delta)$ && $flag$ **then**
10 $N_{f'} \leftarrow N_f, f' \leftarrow f$; //used to record f and N_f for next τ'_s
11 $N_{f'} \leftarrow N_{f'} \uplus (N_{[\tau_e, \tau'_s+\delta]} \setminus N_{[\tau_e,\tau_e]})$; //Lemma 3
12 $\tau_e \leftarrow \tau'_s + \delta, flag \leftarrow$ **false**;
13 `IncreMaxFlow`$^-(f', N_{f'}, \tau_s, \tau'_s, \tau_e, \tau_{r_s}, \tau_{r_e}, r)$; //Section 5.2
14 `IncreMaxFlow`$^+(N_T, f, N_f, \tau_s, \tau_e, \tau'_e, \tau_{r_s}, \tau_{r_e}, r)$;
15 $\tau_e \leftarrow \tau'_e$;
16 $\tau_s \leftarrow \tau'_s, N_f \leftarrow N_{f'}, f \leftarrow f'$;
17 **return** $(r, [\tau_{r_s}, \tau_{r_e}])$; //output binary record of δ-BFlow

Procedure `IncreMaxFlow`$^-(\&f, \&N_f, \tau_s, \tau'_s, \tau_e, \&\tau_{r_s}, \&\tau_{r_e}, \&r)$:
18 $N_f \leftarrow \Delta_{\tau'_s}(N_f), P \leftarrow \emptyset, sum \leftarrow 0$;
19 create and insert into N_f a virtual node $\langle s', \tau_s \rangle$;
20 **foreach** edge $(\langle u, \tau \rangle, \langle u, \tau'_s \rangle), u \neq s$ of N_f **do**
21 create and insert into N_f edge $(\langle u, \tau'_s \rangle, \langle s', \tau_s \rangle)$ with capacity equal to $f(\langle u, \tau \rangle, \langle u, \tau'_s \rangle), sum \leftarrow sum + (\langle u, \tau \rangle, \langle u, \tau'_s \rangle)$;
22 **if** $sum > 0$ **then** // $sum = 0$: no paths exist in P of Lemma 4
23 invoke Dinic on N_f to compute Maxflow from $\langle t, \tau_e \rangle$ to $\langle s', \tau_s \rangle$ and insert the found augmenting paths in set $P', P \leftarrow \{p \mid p$ is the reversed path of $p', p' \in P'\}$;
24 $N_f \leftarrow (N_f \uplus N(P)) \setminus (N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]})$; //Lemma 4
25 `UpdateMaxFlow`$(N_f, f, \tau'_s, \tau_e, \tau_{r_s}, \tau_{r_e}, r)$;

Procedure `UpdateMaxFlow`$(\&N_f, \&f, \tau_s, \tau_e, \&\tau_{r_s}, \&\tau_{r_e}, \&r)$:
26 invoke Dinic on N_f to update f as an MF$[\tau_s, \tau_e]$;
27 **if** $|f|/(\tau_e - \tau_s) > r$ **then**
28 $r \leftarrow |f|/(\tau_e - \tau_s), \tau_{r_s} \leftarrow \tau_s, \tau_{r_e} \leftarrow \tau_e$; //update of δ-BFlow

5.2.3 Optimized δ-BFlow solution by using the incremental Maxflow computation of both the insertion case and the deletion case. Based on BFQ$^+$ and Lemma 4, we further propose an optimized δ-BFlow solution called BFQ*. We put a "*" to BFQ since BFQ* makes use of the incremental Maxflow computation of the insertion case in BFQ$^+$ together with that of the *deletion case*, which computes MF$[\tau_s, \tau_s + \delta]$ for the next smallest timestamp τ_s in Ti(s), and then increase the value of τ_s for the next iteration of the Maxflow computation (Line 3 of Algorithm 1).

As shown in Algorithm 3, BFQ* has the same input and output as those of BFQ and BFQ$^+$. First, after the initialization of the current δ-BFlow in Lines 1-2, Lines 3-5 first invoke Dinic to compute an MF$[\min_{\tau \in T(s)} \{\tau\}, \min_{\tau \in T(s)} \{\tau\} + \delta]$ f with residual network N_f and update the current δ-BFlow in `UpdateMaxFlow`. Based on the current Maxflow f and the residual network N_f *w.r.t.* f, Lines 6-8 iteratively compute Maxflows *w.r.t.* candidate intervals by using the proposed incremental Maxflow computation methods as follows. For each starting timestamp τ_s, Line 14 calls `IncreMaxFlow`$^+$ to conduct incremental Maxflow computation of the insertion case (Algorithm 2). Note that Lines 7-13 record a flow f' and the residual network $N_{f'}$ *w.r.t.* f' used for the Maxflow computation for the next iteration with τ_s in Line 6. Specifically, Lines 10-11 incrementally update $N_{f'}$ by using the $\uplus$ and $\setminus$ operators according Lemma 3. Based on current f' and $N_{f'}$, Line 13 calls `IncreMaxFlow`$^-$ to conduct incremental Maxflow computation of the deletion case (Lemma 4) to compute MF$[\tau'_s, \tau'_s + \delta]$ for the next iteration with τ_s. Finally,



(a) The flow network after conducting the *time-stamp injection* on $N_{f_{[1,4]}}$ for 3, where $N_{f_{[1,4]}}$ is the residual network of the transformed flow network $N_{[1,4]}$ *w.r.t* an MF$[1, 4]$ denoted by $f_{[1,4]}$

(b) The flow network $\left(\Delta_3(N_{f_{[1,4]}} \uplus (N_{[4,5]} \setminus N_{[4,4]}) \uplus N(P))\right) \setminus (N_{[1,3]} \setminus N_{[3,3]})$, where there is no augmenting paths from $\langle s, 3 \rangle$ to $\langle t, 5 \rangle$

Figure 8: [Best viewed in color] Example of the timestamp injection and the incremental Maxflow computation (deletion case) in BFQ* from an MF$[1, 4]$ to an MF$[3, 5]$

after evaluating Maxflows *w.r.t.* all candidate intervals, the flow density and bursting interval are returned as outputs in Line 16.

Regarding `IncreMaxFlow`$^-$ for the incremental Maxflow computation of the deletion case, its detailed procedure is as follows. Given an MF$[\tau_s, \tau_e]$ f, to obtain an MF$[\tau'_s, \tau_e]$ $(\tau'_s > \tau_s)$, Line 18 first conducts the timestamp injection on the residual network N_f *w.r.t.* f for τ'_s by using a traversal on the edges of N_f. Then, Lines 19-23 compute the augmenting paths in set P used in Lemma 4 for withdrawing the flow from f during $[\tau_s, \tau'_s]$. In detail, i) Line 19 creates a virtual node $\langle s', \tau_s \rangle$; ii) for each edge $(\langle u, \tau \rangle, \langle u, \tau'_s \rangle), u \neq s$ of N_f, Line 21 creates and inserts into N_f edge $(\langle u, \tau'_s \rangle, \langle s', \tau_s \rangle)$ with capacity equal to $f(\langle u, \tau \rangle, \langle u, \tau'_s \rangle)$; and iii) Line 23 invokes Dinic to compute Maxflow from $\langle t, \tau_e \rangle$ to $\langle s', \tau_s \rangle$ and inserts into set P the reversed paths of the found augmenting paths. According to Lemma 4, Line 24 computes the flow network $(N_f \uplus N(P)) \setminus (N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]})$ as the updated residual network N_f, and then Line 25 calls `UpdateMaxFlow` on N_f to update flow f as an MF$[\tau'_s, \tau_e]$.

Example 8. *Consider the computation of* 2-BFlow *from s to t in Figure 3(a)'s N_T. Figure 5(c) shows the pattern of the* Maxflow *computation for candidate intervals of* BFQ*. *According to Lemma 4,* BFQ* *computes an* MF$[3, 5]$ *(denoted by $f_{[3,5]}$) in the incremental method (Lines 10-13 of* BFQ*) based on the residual network $N_{f_{[1,4]}}$ w.r.t. $f_{[1,4]}$, where $f_{[1,4]}$ is an* MF$[1, 4]$. *$N_{f_{[1,4]}}$ is shown in Figure 8(a). In detail, Line 11 first computes the flow network $N = N_{f_{[1,4]}} \uplus (N_{[4,5]} \setminus N_{[4,4]})$. Then,* `IncreMaxFlow`$^-$ *(Line 13) computes $f_{[3,5]}$ as follows: i) Line 18 computes $N' = \Delta_3(N)$; ii) Lines 19-21 create and insert into N' the virtual node $\langle s', 1 \rangle$ and edge $(\langle t, 3 \rangle, \langle s', 1 \rangle)$; iii) Line 23 invokes* Dinic *to find the (red) augmenting path p from $\langle t, 4 \rangle$ to $\langle s', 1 \rangle$ in Figure 8(b): $\langle t, 4 \rangle \rightarrow \langle t, 3 \rangle \rightarrow \langle s', 1 \rangle$ having* Flow$(p) = 3$. *As shown by the (red) augmenting path from $\langle t, 4 \rangle$ to $\langle s, 1 \rangle$ in Figure 8(a), p has the same flow as the flow to be withdrawn on edge $(\langle t, 4 \rangle, \langle t, 3 \rangle)$; and iv) based on Lemma 4, Line 24 withdraws the flow on p by computing the flow network $N' \uplus N(\{p\}) \setminus (N_{[1,3]} \setminus N_{[3,3]})$ as shown in Figure 8(b). Since there is no more augmenting paths from $\langle s, 3 \rangle$ to $\langle t, 5 \rangle$ found in the flow network shown in Figure 8(b) (Line 25), we have $|f_{[3,5]}| = |f_{[1,4]}| -$* Flow$(p) = 5 - 3 = 2$. MF$[4, 6]$ *can be computed similarly.*

BFQ*'s correctness is guaranteed by the following lemma and theorem. The detailed proofs of them are presented in Appendix B [1].

Table 2: Statistics of three real world datasets

Dataset	$\|V\|$	$\|E_T\|$	$\|T\|$	Avg. degree	Stddev. degree
BAYC	21K	54.4K	50.2K	5	21.1
Prosper	88K	3.3M	1259	74	141.3
CTU-13	606K	1.7M	22K	5	1550.5
Btc2011	1.99M	3.86M	59K	3	7.9

Lemma 5. `IncreMaxFlow⁻` *computes a set of paths equivalent to set P for the incremental* Maxflow *computation in Lemma 4.*

With Lemmas 4-5, BFQ* can incrementally compute MF$[\tau_s, \tau_e]$s when increasing τ_s's value. As BFQ* evaluates Maxflows *w.r.t.* the same candidate intervals enumerated in BFQ, we have Theorem 2.

Theorem 2. (**Correctness of** BFQ*) *Given a temporal flow network N_T with source s and sink t, and a parameter δ,* BFQ* *returns the flow density and the bursting interval of δ-*BFlow *from s to t.*

Analysis. For a temporal flow network $N_T = (V, E_T, T, C_T)$, Lines 4-5 take $O\Big(\big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$ time. The update of transformed flow networks in Lines 18-21 and Line 14 takes $O\Big(d^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$ time in total. As Lines 6-8 evaluate $O(d^2)$ candidate intervals and Lines 9-16 call $O(1)$ times of Dinic for the incremental Maxflow computation for each candidate interval, the time complexity of Algorithm 3 is $O\Big(d^2 \cdot \big(d_{\max} \cdot |V|\big)^2 \cdot \big(d_{\max} \cdot |V| + |E_T|\big)\Big)$.

6 Experimental Evaluation

We present the experimental settings and then the efficiency evaluation of the proposed δ-BFlow solutions in Section 6.1 and 6.2, respectively. Lastly, we provide a case study in Section 6.3 to demonstrate the application of the proposed δ-BFlow query.

6.1 Experimental Settings

Platform. We implemented in C++ i) some graph data structures for the temporal flow network and the transformed flow network by using the *std::unordered_map* template (an associative container that contains key-value pairs with unique keys, where search, insertion, and removal of elements have average constant-time complexity), which is widely available in graph systems, and then ii) our proposed δ-BFlow solutions by using these structures. The implementationwas made memory-resident. All the evaluations were conducted on a machine with an Intel Core i7-7567U 3.5GHz CPU and 16GB RAM running Ubuntu 20.04.5 LTS. We have also ported our implementation *on top of* a Neo4j backend by using Python and Neo4j's Cypher query language, where all the evaluated δ-BFlow queries can be answered by a *one-off* data export from Neo4j. The time for data export of our largest used dataset was 396 seconds.

Datasets. We used four real-world datasets, namely the *Bitcoin* transaction network[5], a botnet traffic network in *CTU University*[6], the online peer-to-peer loan service of *Prosper Marketplace*[7], and the BAYC transaction dataset utilized from the *non-fungible token (NFT)* repository[8], which are also used in [9, 16, 27, 35, 45]. These datasets are formatted as temporal flow networks in a similar method as that in [27] as follows:

- **Btc2011.** Btc2011 is extracted from the bitcoin transaction network in 2011, where each transaction has a timestamp indicating when this transaction took place. The users and the transactions in the network are considered as the nodes and temporal edges of Btc2011 temporal flow network, respectively, when the transaction amount is considered as the capacities of temporal edges.
- **CTU-13.** CTU-13 is extracted from the botnet traffic network created in CTU university. The IP addresses and the data exchanges with their starting timestamps are considered as the nodes and temporal edges of the CTU-13 temporal flow network, respectively. The data exchange amount is considered as the edge capacities.
- **Prosper.** Prosper is extracted from an online peer-to-peer loan service. The users and the loan transaction are considered as the nodes and temporal edges of the temporal flow network, respectively, when the loan amount is considered as the edge capacities.
- **BAYC.** There are five transaction datasets utilized from the NFT repository [9]. Four out of five dtasets do not have non-trivial flow patterns, specifically, having augmenting paths of a length not less than 2. The only dataset that can be used to investigate the performance is BAYC. Similar to how we formatted Btc2011, we format the BAYC transaction dataset as a temporal flow network.

Note that the timestamps of these formatted temporal flow networks are converted into *sequence numbers* in sequence T. Table 2 summarizes some statistics of these datasets.

Queries. We randomly choose 20 pairs of nodes from the datasets as the source s and the sink t such that there exists non-trivial temporal flows from s to t, which contain paths from s to t having a length not less than 3. As the bursting flow pattern usually happens in a short time, we vary the value of parameter δ for δ-BFlow queries among 3%, 6%, and 9% of $|T|$, where 3% of $|T|$ is set as default.

Solutions. The δ-BFlow solutions to be evaluated are as follows:

- BFQ. We use our proposed δ-BFlow solution in Section 4 for finding the flow density and bursting interval of δ-BFlow by i) enumerating all the candidate intervals, and ii) computing Maxflow on the transformed flow network *w.r.t.* each candidate interval.[9]
- BFQ$^+$. We use the optimization of the incremental Maxflow computation of the insertion case (Section 5.1)[10] on BFQ.
- BFQ*. We further use the optimization of the incremental Maxflow computation of the deletion case (Section 5.2)[10] on BFQ$^+$.

Note that there is a technique to prune nodes and temporal edges that do not affect Maxflows on temporal flow networks [27]. This technique is orthogonal to our solutions that it can be applied to further improve the efficiency. We do not include this technique in our solutions to carefully investigate the performance of our proposed solutions. Moreover, we use the augmenting path based Maxflow algorithm, Dinic, in our δ-BFlow solutions for evaluation.

6.2 Detailed Evaluation on Efficiency

We conducted detailed evaluation to investigate the performance on efficiency of our proposed δ-BFlow solutions. Each query was

[5] https://ieee-dataport.org/open-access/bitcoin-transactions-data-2011-2013
[6] https://www.stratosphereips.org/datasets-ctu13
[7] http://konect.cc/networks
[8] https://livegraphlab.github.io/

[9] As shown in the experimental section of [27], the linear programming (*LP*) cannot handle temporal flow networks with more than 10K edges for computing Maxflows efficiently. As the numbers of temporal edges are large when the candidate intervals are of long lengths, we apply Dinic instead of *LP* to compute Maxflows in BFQ.
[10] We do not apply on BFQ the incremental Maxflow solutions in [28] and [18] since they are designed for dynamic flow networks of inserting or deleting a *single* edge.

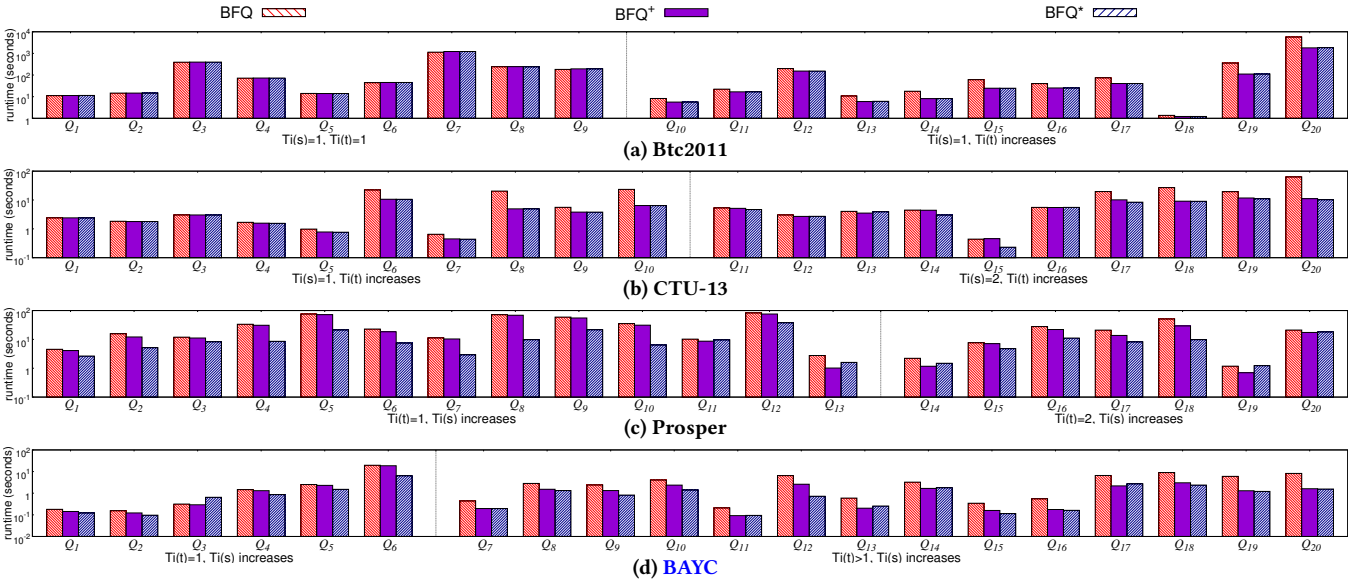


Figure 9: Runtimes of various δ-BFlow solutions on different temporal flow networks (EXP-1)

evaluated 5 times and the average results are reported. Due to the space limitation, the detailed experimental results of EXP-4 and EXP-5 are presented in Appendix C [1].

EXP-1. Overall performance. As shown in Table 2, our proposed methods can reduce at least millions of the $|T|^2$ possible time intervals to at most thousands of the $|d|^2$ candidate intervals. In EXP-1, we investigate the efficiency of our proposed δ-BFlow solutions under the default setting. It can be observed in Figure 9(a) that BFQ$^+$ and BFQ* has similar runtimes on Btc2011 for Q_1-Q_9. The reason is that, for these queries, we have $\mathsf{Ti}(s) = \mathsf{Ti}(t) = 1$ that there is no optimization of the incremental Maxflow computation of both the insertion case and the deletion case. For Q_{10}-Q_{20}, both of BFQ$^+$ and BFQ* run faster than BFQ due to $\mathsf{Ti}(t) > 1$. There exists incremental Maxflow computation of the insertion case. The experimental results are consistent to an observation on the *Bitcoin* transaction network that most of the addresses has few transactions. The runtimes of the δ-BFlow solutions on CTU-13 are similar to those on Btc2011 as shown in Figure 9(b) since the randomly generated queries for CTU-13 have $1 \leq |\mathsf{Ti}(s)| \leq 2$ and $1 \leq |\mathsf{Ti}(t)| \leq 5$.

For Prosper, we see from Figure 9(c) that BFQ* is up to 5x and 3x faster than BFQ and BFQ$^+$, respectively. This is because the queries' sources have tens of out-going edges that there exists much incremental Maxflow computation of the deletion case. Similar results can be observed from Figure 9(d) for BAYC. However, when the network is not large, the runtimes of BFQ* is slightly larger than those of BFQ and BFQ$^+$ for some queries (*e.g.*, Q_{13}, Q_{14} and Q_{19} on Prosper, and Q_3, Q_{13} and Q_{14} on BAYC). This is because the runtime of the network transformation for the incremental Maxflow computation, especially in the deletion case, dominates the runtime.

EXP-2. Speedup of incremental Maxflow **computation.** In EXP-2, we investigate the speedup of BFQ$^+$ and BFQ* for computing Maxflows by using the ratio of BFQ's runtime to BFQ$^+$'s runtime and the ratio of BFQ$^+$'s runtime to BFQ*'s runtime, respectively. To better investigate the incremental Maxflow computation of the

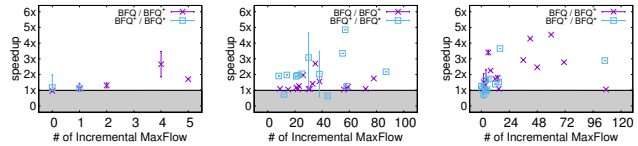


Figure 10: Speedup of BFQ$^+$ and BFQ* when varying the number of times of incremental Maxflow computations (EXP-2)

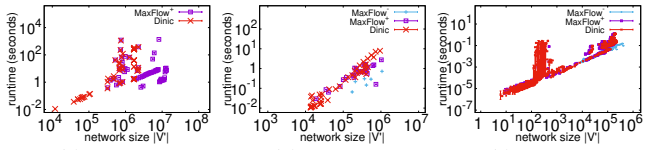


Figure 11: Runtimes of various Maxflow computations when varying the size $|V'|$ of transformed flow network $N = (V', E, C)$ (EXP-3)

proposed solutions, we conducted the speedup evaluation *without* the pruning of Observation 2 on CTU-13, Prosper, and BAYC, where there exists incremental Maxflow computation of both the insertion case and the deletion case. Figure 10 shows the speedup of BFQ$^+$ and BFQ* when varying the number of times of incremental Maxflow computation for finding δ-BFlows for Q_1-Q_{20}. It can be observed from Figure 10(b) that BFQ$^+$ always ran faster than BFQ while BFQ* further ran at most 5x faster than BFQ$^+$.

As our proposed δ-BFlow solutions consist of the network transformation for candidate intervals and the Maxflow computation on transformed flow networks, we report the runtimes of these two parts in the following experiments.

EXP-3. Runtimes of Maxflow **computations when varying the size $|V'|$ of transformed flow network $N = (V', E, C)$.** To compute Maxflow f in N *w.r.t.* a candidate interval, without the network transformation (or the incremental network transformation in Lemmas 3-4), Dinic denotes the computation of f from scratch, while

MaxFlow$^+$ and MaxFlow$^-$ denote the incremental Maxflow computation of the insertion case and the deletion case, respectively. In EXP-3, we conducted BFQ, BFQ$^+$, and BFQ* for Q_1-Q_{20} on Btc2011, CTU-13, and Prosper to show in Figure 11 the runtimes of Dinic, MaxFlow$^+$, and MaxFlow$^-$ on transformed flow networks with different network size $|V'|$.

It can be observed from Figure 11 that Dinic, MaxFlow$^+$, and MaxFlow$^-$ have similar runtimes when $|V'|$ is small that the speedup is not obvious. For Btc2011, we can observe from Figure 11(a) that, there is no cases of MaxFlow$^-$ and the runtime of MaxFlow$^+$ was not larger than that of Dinic. However, there exist some cases of MaxFlow$^+$ conducted on transformed flow networks with large $|V'|$. This is because the δ-BFlow for Q_{20} $(\mathsf{Ti}(t) = 40)$ goes through a dense subgraph of Btc2011, which makes the network transformation of MaxFlow$^+$ (Lemma 3) duplicate more nodes than the network transformation for specific candidate intervals (Section 4.1). For CTU-13, we can see in Figure 11(b) that the runtimes of Dinic, MaxFlow$^+$, and MaxFlow$^-$ increased as $|V'|$ increases while MaxFlow$^-$ is the fastest among the three Maxflow computation on transformed flow networks with the same $|V'|$. For Prosper, Figure 11(c) shows that the runtime of Dinic varies a lot since it's known that the runtime of Dinic for finding Maxflow depends on the topologies of the found augmenting paths, and MaxFlow$^-$ is also the fastest on transformed flow networks having the same $|V'|$.

EXP-4. Runtimes of the network transformations for the proposed δ-BFlow solutions. Let Trans, Trans$^+$, and Trans* denote all the network transformation (from scratch or incrementally) of BFQ, BFQ$^+$ and BFQ*, respectively. We observed from the runtimes of the network transformation on the three evaluated datasets that, when compared to Trans, the runtimes of Trans$^+$ and Trans* had similar trends of speedup as that of the results in Figure 9.

EXP-5. Runtimes of δ-BFlow solutions when varying δ. We observed from the runtimes of our proposed δ-BFlow solutions when varying δ that the runtimes of BFQ for most queries increase as δ increases. This is because a larger δ will make the transformed flow network $N_{[\tau_s,\tau_s+\delta]}$ *w.r.t.* $[\tau_s, \tau_s + \delta]$ have a larger $|V'|$ that finding augmenting paths on $N_{[\tau_s,\tau_s+\delta]}$ takes more time.

We further remark that the runtimes for some queries decrease slightly. For one reason, a larger δ may invalidate some timestamps in $\mathsf{Ti}(t)$ (*i.e.*, τ_e should be not smaller than $\tau_s + \delta$ according to the definition of δ-BFlow (Definition 2)) that fewer candidate intervals were evaluated. For another, the runtimes of BFQ$^+$ and BFQ* are not as sensitive to $|V'|$ as the runtime of BFQ due to incremental Maxflow computation strategies.

6.3 Case Study

To demonstrate the application of δ-BFlow on anomaly detection, we applied BFQ* to find δ-BFlow on the transaction network of a leading technology company. To ensure anonymity, this company is called *Bob*. The transaction network is formatted as a temporal flow network in a way similar to the *Bitcoin* transaction network introduced in Section 6.1. This dataset contains 3.5M nodes as users, 28M temporal edges with 2.5M distinct timestamps as transactions. To simplify the exposition, we converted the timestamps (with a translation conducted by *Bob*) into sequence numbers in T, and evaluated only the transactions having the largest 1% of timestamps in T.[11] As *Bob* had, in fact, labelled a suspicious user set of 184 nodes, we generated δ-BFlow queries as follows: i) we varied $\delta = 0.03$, 0.06 and 0.09 in terms of the number of sequence numbers in T to filter the benign patterns having relatively short time intervals; ii) we inserted into the source set S (*resp.* the sink set T) all the 14 (*resp.* 17) suspicious nodes together with 10 randomly chosen nodes having out-going transactions (*resp.* in-coming transactions); and iii) for each δ, we generated all the combinations of the nodes in S and the nodes in T to obtain $24 \times 27 = 648$ node pairs as δ-BFlow queries.

Table 3: Densities and bursting intervals of found δ-BFlows

δ	Q_1: (s = 1847843, t = 2560590, δ)		Q_2: (s = 2427153, t = 806182, δ)	
	density	bursting interval (corresponding timestamps)	density	bursting interval (corresponding timestamps)
0.03	26,275	[2000-10-31 **19:55:34**, 2000-10-31 **20:26:01**]	74,120	[2000-10-31 23:59:49, 2000-11-03 02:24:13]
0.06	22,140	[2000-10-31 19:54:52, 2000-10-31 20:30:37]	37,036	[2000-10-31 23:59:49, 2000-11-07 15:24:20]
0.09	14,763	[2000-10-31 19:54:52, 2000-10-31 20:49:27]	24,696	[2000-10-31 23:59:49, 2000-11-21 02:36:42]

Among these queries, BFQ* found two interesting queries Q_1 and Q_2. The densities of the found δ-BFlows of Q_1 and Q_2 were significantly larger than the average case. Specifically, Table 3 shows the densities and bursting intervals (in the form of corresponding timestamps) of these δ-BFlows. As *Bob* had, in fact, labelled the source and sink nodes of Q_1 as suspicious users, the finding of δ-BFlow for Q_1 with $\delta = 0.03$ proved that BFQ* detected a bursting transaction transfer pattern between suspicious users that happened in around *half an hour*. For Q_2, the source and the sink were labelled as normal users and the transactions took a long time (at least 3 days), which are common and may not signify an abnormal activity. We can also observe from Table 3 that, a larger δ leads to a smaller density. Therefore, to detect δ-BFlow having a larger burstiness, δ can often be set as relatively small values.

7 Conclusion

In this work, we study a new problem of finding a bursting flow pattern on temporal flow networks. We propose a novel concept of a bursting flow in temporal flow networks, called δ-BFlow, to capture the bursting flow patterns. To find δ-BFlow, we first propose a practical solution called BFQ that transforms temporal flow networks into flow networks so that δ-BFlow can be found by computing the maximum flows in these transformed flow networks. To further improve the efficiency of BFQ, we propose to incrementally compute these maximum flows instead of conducting the Maxflow computation from scratch. The experimental results have shown the efficiency of our proposed δ-BFlow solutions. A case study is provided to show the application of δ-BFlow. As for future work, we plan to study the δ-BFlow query under other practical settings where the δ-BFlow query cannot be handled by our proposed solutions with minor extensions, such as i) finding labeled δ-BFlow in temporal flow networks having keywords on the temporal edges; and ii) δ-BFlow query under a streaming or dynamic model to tackle a more interactive querying on real-time data.

[11] As an application of δ-bursting flow query, we aim to find interesting bursting flow patterns in the most recent periods, such as the recent transactions having the largest 1% timestamps in T.

References

[1] 2024. *Bursting Flow Query on Large Temporal Flow Networks.* Technical Report. https://anonymous.4open.science/r/BurstingFlow-5C8E/bfq2024tr.pdf.

[2] Eleni C. Akrida, Jurek Czyzowicz, Leszek Gasieniec, Lukasz Kuszner, and Paul G. Spirakis. 2019. Temporal flows in temporal networks. *J. Comput. Syst. Sci.* 103 (2019), 46–60.

[3] Albert-Laszlo Barabasi. 2005. The origin of bursts and heavy tails in human dynamics. *Nature* 435 (2005), 207–211.

[4] CERT Advisory CA. 1996. *21 TCP SYN Flooding and IP Spoofing Attacks.* Technical Report. http://www.cert.org/advisories/CA-1996-21.html.

[5] Erin W. Chambers and David Eppstein. 2010. Flows in One-Crossing-Minor-Free Graphs. In *Springer ISAAC*, Vol. 6506. 241–252.

[6] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 2022. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *IEEE FOCS*. 612–623.

[7] Lisi Chen, Shuo Shang, Bin Yao, and Kai Zheng. 2019. Spatio-temporal top-k term search over sliding window. *World Wide Web* 22 (2019), 1953–1970.

[8] Xiaoshuang Chen, Kai Wang, Xuemin Lin, Wenjie Zhang, Lu Qin, and Ying Zhang. 2021. Efficiently Answering Reachability and Path Queries on Temporal Bipartite Graphs. *PVLDB* 14 (2021), 1845–1858.

[9] Yuhang Chen, Jiaxin Jiang, Shixuan Sun, Bingsheng He, and Min Chen. 2024. RUSH: Real-Time Burst Subgraph Detection in Dynamic Graphs. *PVLDB* 17 (2024), 3657–3665.

[10] Lingyang Chu, Yanyan Zhang, Yu Yang, Lanjun Wang, and Jian Pei. 2019. Online Density Bursting Subgraph Detection from Temporal Graphs. *PVLDB* 12 (2019), 2353–2365.

[11] Andrea Fronzetti Colladon and Elisa Remondi. 2017. Using social network analysis to prevent money laundering. *Expert Syst. Appl.* 67 (2017), 49–58.

[12] Efim A Dinic. 1970. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, Vol. 11. 1277–1280.

[13] Lester Randolph Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* (1956), 399–404.

[14] David Gale. 1959. Transient flows in networks. *Michigan Mathematical Journal* (1959), 59–63.

[15] Edoardo Galimberti, Alain Barrat, Francesco Bonchi, Ciro Cattuto, and Francesco Gullo. 2018. Mining (maximal) Span-cores from Temporal Networks. In *ACM CIKM*. 107–116.

[16] Sebastián García, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. *Comput. Secur.* 45 (2014), 100–123.

[17] Andrew V. Goldberg and Robert Endre Tarjan. 1986. A New Approach to the Maximum Flow Problem. In *ACM STOC*. 136–146.

[18] Sergio Greco, Cristian Molinaro, Chiara Pulice, and Ximena Quintana. 2017. Incremental maximum flow computation on evolving networks. In *ACM SAC*. 1061–1067.

[19] Horst W. Hamacher and Stevanus A. Tjandra. 2003. Earliest Arrival Flows with Time-Dependent Data. (2003). https://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-12705

[20] Dorit S. Hochbaum. 2008. The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem. *Oper. Res.* 56 (2008), 992–1009.

[21] Jan M. Hochstein and Karsten Weihe. 2007. Maximum s-t-flow with k crossings in O(k3n log n) time. In *ACM SODA*. 843–847.

[22] Petter Holme. 2014. Temporal Networks. In *Encyclopedia of Social Network Analysis and Mining*. 2119–2129.

[23] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. FRAUDAR: Bounding Graph Fraud in the Face of Camouflage. In *ACM KDD*. 895–904.

[24] Silu Huang, Ada Wai-Chee Fu, and Ruifeng Liu. 2015. Minimum Spanning Trees in Temporal Graphs. In *ACM SIGMOD*. 419–430.

[25] Wei Huang, Yongqing Wang, and Liehuang Zhu. 2023. A Time Impulse Neural Network Framework for Solving the Minimum Path Pair Problems of the Time-Varying Network. *IEEE TKDE* 35 (2023), 7681–7692.

[26] Jiaxin Jiang, Yuan Li, Bingsheng He, Bryan Hooi, Jia Chen, and Johan Kok Zhi Kang. 2022. Spade: A Real-Time Fraud Detection Framework on Evolving Graphs. *PVLDB* 16 (2022), 461–469.

[27] Chrysanthi Kosyfaki, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2021. Flow Computation in Temporal Interaction Networks. In *IEEE ICDE*. 660–671.

[28] S. Kumar and P. Gupta. 2003. An Incremental Algorithm for the Maximum Flow Problem. *J. Math. Model. Algorithms* 2 (2003), 1–16.

[29] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent Community Search in Temporal Networks. In *IEEE ICDE*. 797–808.

[30] Zihan Luo, Lei Li, Mengxuan Zhang, Wen Hua, Yehong Xu, and Xiaofang Zhou. 2022. Diversified Top-k Route Planning in Road Network. *PVLDB* 15 (2022), 3199–3212.

[31] Malte Möser, Rainer Böhme, and Dominic Breuker. 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. In *IEEE eCrime*. 1–14.

[32] Hongchao Qin, Rong-Hua Li, Guoren Wang, Lu Qin, Yurong Cheng, and Ye Yuan. 2019. Mining Periodic Cliques in Temporal Networks. In *IEEE ICDE*. 1130–1141.

[33] Hongchao Qin, Ronghua Li, Ye Yuan, Guoren Wang, Lu Qin, and Zhiwei Zhang. 2022. Mining Bursting Core in Large Temporal Graph. *PVLDB* 15 (2022), 3911–3923.

[34] Melanie Schmidt and Martin Skutella. 2014. Earliest arrival flows in networks with multiple sinks. *Discret. Appl. Math.* 164 (2014), 320–327.

[35] Omer Shafiq. 2019. Bitcoin Transactions Data 2011-2013. https://doi.org/10.21227/8dfs-0261

[36] Daniel Dominic Sleator and Robert Endre Tarjan. 1985. Self-Adjusting Binary Search Trees. *J. ACM* 32 (1985), 652–686.

[37] Daniel Dominic Kaplan Sleator. 1981. *An o(nm log n) algorithm for maximum network flow.* Ph. D. Dissertation.

[38] Sibo Wang, Wenqing Lin, Yi Yang, Xiaokui Xiao, and Shuigeng Zhou. 2015. Efficient Route Planning on Public Transportation Networks: A Labelling Approach. In *ACM SIGMOD*. 967–982.

[39] Yong Wang, Guoliang Li, and Nan Tang. 2019. Querying Shortest Paths on Time Dependent Road Networks. *PVLDB* 12 (2019), 1249–1261.

[40] Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications.* Cambridge University Press.

[41] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. 2019. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *CoRR* abs/1908.02591 (2019).

[42] Dong Wen, Yilun Huang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2020. Efficiently Answering Span-Reachability Queries in Large Temporal Graphs. In *IEEE ICDE*. 1153–1164.

[43] Dong Wen, Bohua Yang, Ying Zhang, Lu Qin, Dawei Cheng, and Wenjie Zhang. 2022. Span-reachability querying in large temporal graphs. *VLDBJ* 31 (2022), 629–647.

[44] W. L. Wilkinson. 1971. An Algorithm for Universal Maximal Dynamic Flows in a Network. *Oper. Res.* 19 (1971), 1602–1612.

[45] Gavin Wood. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum* (2014). https://ethereum.github.io/yellowpaper/paper.pdf

[46] Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. 2014. Path Problems in Temporal Graphs. *PVLDB* 7 (2014), 721–732.

[47] Huanhuan Wu, Yuzhen Huang, James Cheng, Jinfeng Li, and Yiping Ke. 2016. Reachability and time-based path queries in temporal graphs. In *IEEE ICDE*. 145–156.

[48] Jiajing Wu, Jieli Liu, Weili Chen, Huawei Huang, Zibin Zheng, and Yan Zhang. 2022. Detecting Mixing Services via Mining Bitcoin Transaction Network With Hybrid Motifs. *IEEE Trans. Syst. Man Cybern. Syst.* 52 (2022), 2237–2249.

[49] Haoxuan Xie, Yixiang Fang, Yuyang Xia, Wensheng Luo, and Chenhao Ma. 2023. On Querying Connected Components in Large Temporal Graphs. *Proc. ACM Manag. Data* 1 (2023), 170:1–170:27.

[50] Hai Yang and Michael G. H. Bell. 1998. Models and algorithms for road network design: a review and some new developments. *Transport Reviews* 18 (1998), 257–278.

[51] Yajun Yang, Jeffrey Xu Yu, Hong Gao, Jian Pei, and Jianzhong Li. 2014. Mining most frequently changing component in evolving graphs. *World Wide Web* 17 (2014), 351–376.

[52] Michael Yu, Dong Wen, Lu Qin, Ying Zhang, Wenjie Zhang, and Xuemin Lin. 2021. On Querying Historical K-Cores. *PVLDB* 14 (2021), 2033–2045.

[53] Ye Yuan, Xiang Lian, Guoren Wang, Yuliang Ma, and Yishu Wang. 2019. Constrained Shortest Path Query in a Large Time-Dependent Graph. *PVLDB* 12 (2019), 1058–1070.

[54] Qianzhen Zhang, Deke Guo, Xiang Zhao, Long Yuan, and Lailong Luo. 2023. Discovering Frequency Bursting Patterns in Temporal Graphs. In *IEEE ICDE*. 599–611.

[55] Tianming Zhang, Yunjun Gao, Lu Chen, Wei Guo, Shiliang Pu, Baihua Zheng, and Christian S. Jensen. 2019. Efficient distributed reachability querying of massive temporal graphs. *VLDBJ* 28 (2019), 871–896.

[56] Tianming Zhang, Yunjun Gao, Linshan Qiu, Lu Chen, Qingyuan Linghu, and Shiliang Pu. 2020. Distributed time-respecting flow graph pattern matching on temporal graphs. *World Wide Web* 23 (2020), 609–630.

Table 4: Summary of time complexities of some existing Maxflow algorithms on a flow network $N = (V, E, C)$ where V, E, and C are the set of nodes, the set of edges, and the capacity mapping function, respectively. $|f|$ denotes the flow value of the maximum flow f in N.

Algorithm	Network type	Time complexity
Ford-Fulkerson [13]	traditional network	$O(\|E\|\cdot\|f\|)$
Dinic [12]	traditional network	$O(\|V\|^2\cdot\|E\|)$
Push-relabel [17]	traditional network	$O(\|V\|^3)$ or $O\big(\|V\|\cdot\|E\|\cdot\log(\|V\|^2/\|E\|)\big)$
Pseudoflow [20]	traditional network	$O(\|V\|\cdot\|E\|\cdot\log\|V\|)$
Hochstein & Weihe [21]	k-crossings network	$O(k^3\cdot\|V\|\cdot\log\|V\|)$
Chambers & Eppstein [5]	one-crossing-minor-free network	$O(\|V\|\cdot\log\|V\|)$

A Time Complexities of Some Existing Maxflow Algorithms

Table 4 summarizes the time complexities of some existing Maxflow algorithms on different types of networks.

B Proofs

In this appendix, we present the detailed proofs of the lemmas and theorems of this paper.

Lemma 1. (**Equivalent flow of the same flow value**) *Consider a temporal flow network N_T, a source s at the starting timestamp τ_s, a sink t at the ending timestamp τ_e, and the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T w.r.t. $[\tau_s,\tau_e]$. Given any temporal flow F from s to t in N_T, there exists a flow f from node $\langle s,\tau_s\rangle$ to node $\langle t,\tau_e\rangle$ in $N_{[\tau_s,\tau_e]}$ such that $|f| = |F|$, and vice versa.*

Proof. We prove Lemma 1 by constructing the target f and F from any temporal flow F and any flow f, respectively.

Construction: $F \Rightarrow f$. Given F from s at τ_s to t at τ_e in $N_T = (V, E_T, T, C_T)$, the flow f in $N_{[\tau_s,\tau_e]} = (V', E, C)$ can be constructed as follows:

1) $\forall\langle u,\tau\rangle, \langle v,\tau\rangle \in V'$, if $F(u,v,\tau) > 0$, we set $f\big(\langle u,\tau\rangle,\langle v,\tau\rangle\big) = F(u,v,\tau)$. Otherwise, $f\big(\langle u,\tau\rangle,\langle v,\tau\rangle\big) = 0$;
2) let $\mathsf{Ti}_{[\tau_s,\tau_e]}(u)$ denote the set $\{\tau \mid \tau \in \mathsf{Ti}(u), \tau_s \le \tau \le \tau_e\}$ and let τ_i denote the i^{th} timestamp in $\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(u)\big)$. Then, $\forall\langle u,\tau_i\rangle$, $\langle u,\tau_{i+1}\rangle \in V'$ and $u \ne s$,
   - $f\big(\langle u,\tau_1\rangle,\langle u,\tau_2\rangle\big) = \sum_{v\in V} F(v,u,\tau_1) - \sum_{v\in V} F(u,v,\tau_1)$; and
   - $f\big(\langle u,\tau_i\rangle,\langle u,\tau_{i+1}\rangle\big) = f\big(\langle u,\tau_{i-1}\rangle,\langle u,\tau_i\rangle\big) + \sum_{v\in V} F(v,u,\tau_i) - \sum_{v\in V} F(u,v,\tau_i)$, $i \in \Big[2, \big|\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(u)\big)\big| - 1\Big]$;
3) let $n = \big|\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(s)\big)\big|$ and let τ_i denote the i^{th} timestamp in $\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(s)\big)$. Then, $\forall\langle s,\tau_{i-1}\rangle, \langle s,\tau_i\rangle \in V'$,
   - $f\big(\langle s,\tau_{n-1}\rangle,\langle s,\tau_n\rangle\big) = \sum_{v\in V} F(s,v,\tau_n)$; and
   - $f\big(\langle s,\tau_{i-1}\rangle,\langle s,\tau_i\rangle\big) = f\big(\langle s,\tau_i\rangle,\langle s,\tau_{i+1}\rangle\big) + \sum_{v\in V} F(s,v,\tau_i)$, $i \in [2, n-1]$;
4) $\forall\ \langle u,\tau\rangle, \langle v,\tau'\rangle \in V'$ $(\tau \ne \tau')$ such that $\big(\langle u,\tau\rangle,\langle v,\tau'\rangle\big) \notin E$,
$$0 \le f\big(\langle u,\tau\rangle,\langle v,\tau'\rangle\big) \le C\big(\langle u,\tau\rangle,\langle v,\tau'\rangle\big) = 0 \Rightarrow f\big(\langle u,\tau\rangle,\langle v,\tau'\rangle\big) = 0$$

It can be easily proved that i) f satisfies the capacity constraint and the flow conservation of a flow (Section 3), and ii) $|f| = |F|$.

Construction: $f \Rightarrow F$. Given f from $\langle s,\tau_s\rangle$ to $\langle t,\tau_e\rangle$ in $N_{[\tau_s,\tau_e]} = (V', E, C)$ of $N_T = (V, E_T, T, C_T)$, the temporal flow F in N_T can be constructed as follows: $\forall u, v \in V$, $\tau \in [\tau_s,\tau_e]$, if $\langle u,\tau\rangle, \langle v,\tau\rangle \in V'$, $F(u,v,\tau) = f\big(\langle u,\tau\rangle,\langle v,\tau\rangle\big)$; otherwise, $F(u,v,\tau) = 0$. F satisfies the following:

1) the capacity constraint. $\forall u, v \in V$, $\tau \in [\tau_s,\tau_e]$, if $\langle u,\tau\rangle, \langle v,\tau\rangle \in V'$, $F(u,v,\tau) = f\big(\langle u,\tau\rangle,\langle v,\tau\rangle\big) \le C\big(\langle u,\tau\rangle,\langle v,\tau\rangle\big) = C_T(u,v,\tau)$; otherwise, $F(u,v,\tau) = 0 \le C_T(u,v,\tau)$.
2) the flow conservation (Equation (3)). Given a node u of N_T, let $n = \big|\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(u)\big)\big|$ and let τ_i denote the i^{th} timestamp in $\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(u)\big)$. $\forall u \in V - \{s,t\}$, $i,j \in [2,n]$ $(j > i)$, if $j \ne i+1$, $0 \le f\big(\langle u,\tau_i\rangle,\langle u,\tau_j\rangle\big) \le C\big(\langle u,\tau_i\rangle,\langle u,\tau_j\rangle\big) = 0$. As the flow conservation holds for f, $\forall u \in V - \{s,t\}$, we have,
$$\sum_{i\in[1,n]}\sum_{\langle v,\tau_i\rangle\in V'} f\big(\langle v,\tau_i\rangle,\langle u,\tau_i\rangle\big) + \sum_{i\in[2,n]} f\big(\langle u,\tau_{i-1}\rangle,\langle u,\tau_i\rangle\big) = \sum_{i\in[1,n]}\sum_{\langle v,\tau_i\rangle\in V'} f\big(\langle u,\tau_i\rangle,\langle v,\tau_i\rangle\big) + \sum_{i\in[1,n-1]} f\big(\langle u,\tau_i\rangle,\langle u,\tau_{i+1}\rangle\big)$$
$$\Rightarrow \sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]} F(v,u,\tau) = \sum_{v\in V}\sum_{\tau\in[\tau_s,\tau_e]} F(u,v,\tau)$$
3) the time constraint (Equation (4)). $\forall u \in V - \{s,t\}$, $\tau' \in [\tau_s,\tau_e]$, let τ_j denote the j^{th} timestamp in $\mathsf{Seq}\big(\mathsf{Ti}_{[\tau_s,\tau_e]}(u)\big)$ such that $\tau_j \le \tau' < \tau_{j+1}$. As the flow conservation holds for f on the nodes $\langle u,\tau_1\rangle$, .., $\langle u,\tau_j\rangle$ of $N_{[\tau_s,\tau_e]}$, we have,
$$\sum_{i\in[1,j]}\sum_{\langle v,\tau_i\rangle\in V'} f\big(\langle v,\tau_i\rangle,\langle u,\tau_i\rangle\big) + \sum_{i\in[2,j]} f\big(\langle u,\tau_{i-1}\rangle,\langle u,\tau_i\rangle\big) = \sum_{i\in[1,j]}\sum_{\langle v,\tau_i\rangle\in V'} f\big(\langle u,\tau_i\rangle,\langle v,\tau_i\rangle\big) + \sum_{i\in[1,j]} f\big(\langle u,\tau_i\rangle,\langle u,\tau_{i+1}\rangle\big)$$
$$\Rightarrow \sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']} F(v,u,\tau) = \Big(\sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']} F(u,v,\tau)\Big) + f\big(\langle u,\tau_j\rangle,\langle u,\tau_{j+1}\rangle\big)$$
$$\Rightarrow \sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']} F(v,u,\tau) \ge \sum_{v\in V}\sum_{\tau\in[\tau_s,\tau']} F(u,v,\tau)$$

With the analyses above, F is a temporal flow in N_T that $|F| = |f|$.

Putting the two constructions (from F to f, and from f to F) together, Lemma 1 holds. □

Lemma 2. *Given a δ-BFlow query $Q = (s,t,\delta)$ on a temporal flow network N_T, δ-BFlow in N_T for Q can be found by finding $\mathsf{MF}[\tau_s,\tau_e]$, where τ_s and τ_e satisfy either*

- *$\tau_e = \tau_s + \delta$, $\tau_s \in \mathsf{Ti}(s)$; or*
- *$[\tau_s,\tau_e]$ is a core interval such that $\tau_e - \tau_s > \delta$.*

Proof. For a δ-BFlow query Q, *if $[\tau_s,\tau_e]$ is a bursting interval*, there are two cases of $[\tau_s,\tau_e]$.

Case 1: $[\tau_s,\tau_e]$ is a core interval. In this case, $\mathsf{MF}[\tau_s,\tau_e]$ can be found by enumerating as candidate intervals *all* the $O(d^2)$ core intervals having the interval length larger than δ.

Case 2: $[\tau_s,\tau_e]$ is not a core interval. As $[\tau_s,\tau_e]$ is a bursting interval, there must exist a core interval $[\tau'_s,\tau'_e]$ such that:

- $[\tau'_s,\tau'_e]$ is a subinterval of $[\tau_s,\tau_e]$; and
- $\mathsf{MF}[\tau'_s,\tau'_e]$ and $[\tau_s,\tau_e]$ have the same flow value.

If $[\tau'_s,\tau'_e]$ does not exist, there is no temporal flow during $[\tau_s,\tau_e]$ in N_T, which is *contradictory* to the assumption that $[\tau_s,\tau_e]$ is a bursting interval. Moreover, we have $\tau'_e - \tau'_s < \delta$. The reason is that, if $\tau'_e - \tau'_s \ge \delta$, $[\tau_s,\tau_e]$ cannot a bursting interval since $\mathsf{MF}[\tau'_s,\tau'_e]$ and $[\tau_s,\tau_e]$ have the same flow value while $[\tau'_s,\tau'_e]$ has a smaller interval length. As $N_{[\tau'_s,\tau'_e]}$ is a subgraph of the transformed flow networks $N_{[\tau,\tau+\delta]}$, $\tau'_e - \delta \le \tau \le \tau'_s$, we can know that Maxflows *w.r.t.* all the time intervals $[\tau,\tau+\delta]$ have the same flow value as that of $\mathsf{MF}[\tau_s,\tau_e]$. Therefore, we can find only $\mathsf{MF}[\tau'_s,\tau'_s+\delta]$, and then

compute the density $\left|\mathsf{MF}[\tau_s', \tau_s' + \delta]\right|/\delta$ and the bursting intervals $[\tau, \tau + \delta]$, $\tau_e' - \delta \le \tau \le \tau_s'$ as the query results.[12] Such procedure can be achieved by enumerating $|\mathsf{Ti}(s)|$ time intervals $[\tau', \tau' + \delta]$, $\tau' \in \mathsf{Ti}(s)$ as candidate intervals.

Based on the above analysis, we can enumerate only the $O(d^2)$ candidate intervals illustrated in Lemma 2 to find δ-BFlow. Hence, Lemma 2 holds. □

Lemma 3. (**Incremental** Maxflow (**insertion case**)) *Consider a temporal flow network N_T with source s and sink t, the transformed flow network $N_{[\tau,\tau_e]}$ of N_T w.r.t. $[\tau, \tau_e]$, and the following notations.*

- *$f_{[\tau_s,\tau_e]}$ denotes an $\mathsf{MF}[\tau_s, \tau_e]$ (where $\tau_s \ge \tau$) in $N_{[\tau,\tau_e]}$, obtained by* Dinic *by finding augmenting paths; and*
- *τ_e' is a timestamp larger than τ_e.*

With these notations, a Maxflow *$f_{[\tau_s,\tau_e']}$ w.r.t. $[\tau_s, \tau_e']$ in $N_{[\tau,\tau_e']}$ can be obtained by finding augmenting paths $p_{\tau_e'}$s from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ on the flow network $N_{f_{[\tau_s,\tau_e]}} \uplus \left(N_{[\tau_e,\tau_e']} \setminus N_{[\tau_e,\tau_e]}\right)$ having*

$$|f_{[\tau_s,\tau_e']}| = |f_{[\tau_s,\tau_e]}| + \sum_{p_{\tau_e'}s} \mathsf{Flow}(p_{\tau_e'}),$$

where $N_{f_{[\tau_s,\tau_e]}}$ is the residual network of $N_{[\tau,\tau_e]}$ w.r.t. $f_{[\tau_s,\tau_e]}$.

Proof. We prove Lemma 3 by showing that the found flow in $N_{f_{[\tau_s,\tau_e]}} \uplus \left(N_{[\tau_e,\tau_e']} \setminus N_{[\tau_e,\tau_e]}\right)$ is identical to a Maxflow from source $\langle s, \tau_s\rangle$ to sink $\langle t, \tau_e'\rangle$ in $N_{[\tau,\tau_e']}$. Let's first consider the transformation of flow network $N' = N_{[\tau,\tau_e]} \uplus \left(N_{[\tau_e,\tau_e']} \setminus N_{[\tau_e,\tau_e]}\right)$. There exist three exhausted cases of any node $\langle u, \tau_e\rangle$ of N':

- **Case 1:** $\langle u, \tau_e\rangle$ has no in-coming edges. For this case, we insert into N' node $\langle u, \tau\rangle$ and edge $\left(\langle u, \tau\rangle, \langle u, \tau_e\rangle\right)$ with capacity equal to $+\infty$;
- **Case 2:** $\langle u, \tau_e\rangle$ has only one in-coming edge $\left(\langle u, \tau'\rangle, \langle u, \tau_e\rangle\right)$, $\tau' < \tau_e$ and one out-going edge $\left(\langle u, \tau_e\rangle, \langle u, \tau''\rangle\right)$, $\tau'' > \tau_e$. For this case, we remove node $\langle u, \tau_e\rangle$ and these two edges, and then insert into N' an edge $\left(\langle u, \tau'\rangle, \langle u, \tau''\rangle\right)$ with capacities $+\infty$;
- **Case 3.** Case 3 corresponds to the rest cases, where no transformation need to be conducted.

We can easily verify that i) such a transformation transforms N' into $N_{[\tau,\tau_e']}$, according to the network transformation procedure (Section 4.1) and the operators $\uplus$ and $\setminus$ (Section 5.1); and ii) such a transformation has *no effect* on a Maxflow f' from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ since f' does not go through the inserted edge in Case 1 and f' goes through the same flow from $\langle u, \tau'\rangle$ to $\langle u, \tau''\rangle$ in Case 2. Hence, a Maxflow f' from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ in N' can be transformed into a Maxflow f *w.r.t.* $[\tau_s, \tau_e']$ in $N_{[\tau,\tau_e']}$ by using the above transformation. That is, we have $|f'| = |f|$.

To compute a Maxflow f' in N', we can use existing methods [12, 13] that iteratively finds augmenting paths from residual networks as follows. We first compute the set P of the augmenting paths from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e\rangle$ in $N_{[\tau,\tau_e]}$ to obtain an $\mathsf{MF}[\tau_s, \tau_e]$ (denoted by $f_{[\tau_s,\tau_e]}$) and the corresponding residual network $N_{f_{[\tau_s,\tau_e]}}$. Note that each path p in P can be transformed into a valid augmenting path from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ in N' with the same flow value $\mathsf{Flow}(p)$ by assigning flow value $\mathsf{Flow}(p)$ on the edges constructed by the timestamp inlining on t for $[\tau_e, \tau_e']$. We can also note that $N_{f_{[\tau_s,\tau_e]}} \uplus \left(N_{[\tau_e,\tau_e']} \setminus N_{[\tau_e,\tau_e]}\right)$ is identical to the residual network of N' *w.r.t.* these transformed augmenting paths. The reason is that these augmenting paths go through only the edges of $N_{[\tau,\tau_e]}$ and the edges constructed by the timestamp inlining on t for $[\tau_e, \tau_e']$ whose capacities equal to $+\infty$, which do not affect the topology of the residual network *w.r.t.* the flow of these augmenting paths.

Therefore, we can continue to find augmenting paths $p_{\tau_e'}$s from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ in the identical residual network $N_{f_{[\tau_s,\tau_e]}} \uplus \left(N_{[\tau_e,\tau_e']} \setminus N_{[\tau_e,\tau_e]}\right)$. The paths in P with a transformation together with the paths $p_{\tau_e'}$s are valid augmenting paths in N', and there are no more augmenting paths from $\langle s, \tau_s\rangle$ to $\langle t, \tau_e'\rangle$ in the residual network of N' *w.r.t.* the flow of these paths. Having the fact that "*a flow f from s to t is maximum if and only if there is no augmenting path; that is, t is not reachable from s in the residual graph*" [17], we obtain an $\mathsf{MF}[\tau_s, \tau_e']$ (denoted by f') in N' and hence, an $\mathsf{MF}[\tau_s, \tau_e']$ (denoted by $f_{[\tau_s,\tau_e']}$) in $N_{[\tau,\tau_e']}$ having $|f_{[\tau_s,\tau_e']}| = |f'| = \sum_{p\in P} \mathsf{Flow}(p) + \sum_{p_{\tau_e'}s} \mathsf{Flow}(p_{\tau_e'}) = |f_{[\tau_s,\tau_e]}| + \sum_{p_{\tau_e'}s} \mathsf{Flow}(p_{\tau_e'})$. Therefore, Lemma 3 holds. □

Theorem 1. *Given a temporal flow network N_T with source s and sink t, and a parameter δ,* BFQ^+ *returns the flow density and the bursting interval of δ-*BFlow *from s to t.*

Proof. As introduced in Section 4, BFQ computes the flow density and the bursting interval of δ-BFlow by computing Maxflows *w.r.t.* all the candidate intervals. Therefore, we prove Theorem 1 by showing that BFQ^+ also computes Maxflows *w.r.t.* the same candidate intervals enumerated in BFQ.

For the candidate intervals $[\tau_s, \tau_s + \delta]$, $\tau_s \in \mathsf{Ti}(s)$ enumerated in Lines 3-4 of BFQ, BFQ^+ computes Maxflows *w.r.t.* these intervals by using Dinic in Lines 3-6. Regarding the rest candidate intervals $[\tau_s, \tau_e]$, $\tau_s \in \mathsf{Ti}(s)$, $\tau_e \in \mathsf{Ti}(t)$ enumerated in Lines 3-4 of BFQ for the incremental Maxflow computation of the insertion case, BFQ^+ computes Maxflows *w.r.t.* these intervals by using the incremental procedure `IncreMaxFlow`$^+$ in Lines 9-10. Note that `IncreMaxFlow`$^+$ can incrementally compute Maxflows according to Lemma 3. Therefore, BFQ^+ and BFQ compute Maxflows *w.r.t.* the same candidate intervals. Theorem 1 holds. □

Lemma 4. (**Incremental** Maxflow (**deletion case**)) *Consider a temporal flow network N_T with source s and sink t, the transformed flow network $N_{[\tau_s,\tau_e]}$ of N_T w.r.t. $[\tau_s, \tau_e]$, a timestamp $\tau_s' \in \mathsf{Ti}(s)$ that $\tau_s < \tau_s' < \tau_e$, and the following notations.*

- *$f_{[\tau_s,\tau_e]}$ denotes an $\mathsf{MF}[\tau_s, \tau_e]$ in $N_{[\tau_s,\tau_e]}$ obtained by* Dinic *by finding augmenting paths; and*
- *P denotes the set of the found augmenting paths that do not pass through node $\langle s, \tau_s'\rangle$.*

With these notations, a Maxflow *$f_{[\tau_s',\tau_e]}$ w.r.t. $[\tau_s', \tau_e]$ in $N_{[\tau_s',\tau_e]}$ can be obtained by finding augmenting paths $p_{\tau_s'}$s from $\langle s, \tau_s'\rangle$ to $\langle t, \tau_e\rangle$ in $\left(\Delta_{\tau_s'}\left(N_{f_{[\tau_s,\tau_e]}} \uplus N(P)\right)\right) \setminus \left(N_{[\tau_s,\tau_s']} \setminus N_{[\tau_s',\tau_s']}\right)$ having,*

$$|f_{[\tau_s',\tau_e]}| = |f_{[\tau_s,\tau_e]}| + \sum_{p_{\tau_s'}s} \mathsf{Flow}(p_{\tau_s'}) - \sum_{p\in P} \mathsf{Flow}(p),$$

where $N_{f_{[\tau_s,\tau_e]}}$ is the residual network of $N_{[\tau_s,\tau_e]}$ w.r.t. $f_{[\tau_s,\tau_e]}$.

Proof. We prove Lemma 4 by showing that the found flow in Lemma 4 is identical to a Maxflow from source $\langle s, \tau_s'\rangle$ to sink $\langle t, \tau_e\rangle$

[12] τ_s' (*resp.* τ_e') can be easily obtained when computing an $\mathsf{MF}[\tau_s', \tau_s' + \delta]$ f by recording the smallest timestamp τ such that $f\left(\langle s, \tau\rangle, \langle v, \tau\rangle\right) > 0$, $v \ne s$ (*resp.* the largest timestamp τ such that $f\left(\langle v, \tau\rangle, \langle t, \tau\rangle\right) > 0$, $v \ne t$).

in $N_{[\tau'_s,\tau_e]}$. Let's first consider the flow network $N' = \left(\Delta_{\tau'_s}\left(N_{[\tau_s,\tau_e]}\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$. According to the definition of timestamp injection (Section 5.2), we can easily know that i) $N' = N_{[\tau'_s,\tau_e]}$; and ii) the timestamp injection and operator $\setminus$ for N' have *no effect* on a Maxflow f' *w.r.t.* $[\tau'_s,\tau_e]$ in N' since f' does not go through the inserted or deleted edges. Hence, f' is also a Maxflow f *w.r.t.* $[\tau'_s,\tau_e]$ in $N_{[\tau'_s,\tau_e]}$ and we have $|f'| = |f|$.

f' can be computed by using existing methods [12, 13] that iteratively finds augmenting paths from residual networks. Assume that we compute an $\mathsf{MF}[\tau_s,\tau_e]$ (denoted by $f_{[\tau_s,\tau_e]}$) in $N_{[\tau_s,\tau_e]}$. For $f_{[\tau_s,\tau_e]}$, we insert into set P the found augmenting paths that do not go through $\langle s,\tau'_s\rangle$, and insert the rest found augmenting paths into set P'. Note that the subpaths of the paths in P' starting from $\langle s,\tau'_s\rangle$ to $\langle t,\tau_e\rangle$ can also be augmenting paths for an $\mathsf{MF}[\tau'_s,\tau_e]$ (denoted by f') in N' having the same flow values on them. We insert these subpaths into set P''. According to the definitions of residual network (Section 3.1) and augmenting flow network (Section 5.2), it can be deduced that $\left(\Delta_{\tau'_s}\left(N_{f_{[\tau_s,\tau_e]}} \uplus N(P)\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$ is identical to the residual network of N' *w.r.t.* the augmenting paths in P'' since the flow values of augmenting paths in P are withdrawn by conducting operator $\uplus$ between $N_{f_{[\tau_s,\tau_e]}}$ and the augmenting flow network of P. Then, we continue to find augmenting paths $p_{\tau'_s}$s from $\langle s,\tau'_s\rangle$ to $\langle t,\tau_e\rangle$ in this identical residual network $\left(\Delta_{\tau'_s}\left(N_{f_{[\tau_s,\tau_e]}} \uplus N(P)\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$. The paths $p_{\tau'_s}$s together with the paths in P'' are valid augmenting paths in N', and there are no more augmenting paths from $\langle s,\tau'_s\rangle$ to $\langle t,\tau_e\rangle$ in the residual network of N' *w.r.t.* these paths. Having the fact that "*a flow f from s to t is maximum if and only if there is no augmenting path; that is, t is not reachable from s in the residual graph*" [17], we obtain an $\mathsf{MF}[\tau'_s,\tau_e]$ (denoted by f') in N' and hence, an $\mathsf{MF}[\tau'_s,\tau_e]$ (denoted by $f_{[\tau'_s,\tau_e]}$) in $N_{[\tau'_s,\tau_e]}$ having $|f_{[\tau'_s,\tau_e]}| = |f'| = \sum_{p\in P''} \mathsf{Flow}(p) + \sum_{p_{\tau'_s}\text{s}} \mathsf{Flow}(p_{\tau'_s}) = |f_{[\tau_s,\tau_e]}| - \sum_{p\in P} \mathsf{Flow}(p) + \sum_{p_{\tau'_s}\text{s}} \mathsf{Flow}(p_{\tau'_s})$. Therefore, Lemma 3 holds. □

Lemma 5. `IncreMaxFlow⁻` *computes a set of paths equivalent to set P for the incremental* Maxflow *computation in Lemma 4.*

Proof. We prove Lemma 5 by 1) converting the proof into the proof of an equation, and then 2) showing that this equation holds.

Step 1). Consider a temporal flow network N_T with source s and sink t. Given a time interval $[\tau_s,\tau_e]$ and a timestamp $\tau'_s \in \mathsf{Ti}(s)$ such that $\tau_s < \tau'_s \le \tau_e$, assume that Dinic computes an $\mathsf{MF}[\tau_s,\tau_e]$ (denoted by f) in $N_{[\tau_s,\tau_e]}$ with a residual network N_f and finds a set of augmenting paths for f. This set can be divided into two subsets P and P', where i) P contains the paths that do not pass through node $\langle s,\tau'_s\rangle$, (*i.e.*, P is the set for the incremental Maxflow computation in Lemma 4), and ii) P' contains the paths that pass through node $\langle s,\tau'_s\rangle$.

To prove Lemma 5, we need to prove that the set of paths computed by `IncreMaxFlow⁻` is equivalent to P. That is, *by using* `IncreMaxFlow⁻`*, the found augmenting paths from $\langle t,\tau_e\rangle$ to the created virtual node $\langle s',\tau_s\rangle$ can withdraw the same flow value from each edge of $N_f \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$ as the paths in P do.* Note that there is no need to withdraw flows on all the edges of $N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}$ since the corresponding temporal edges of these edges have timestamps smaller than τ'_s that these edges cannot contribute to $\mathsf{MF}[\tau'_s,\tau_e]$, and hence, cannot contribute to incremental Maxflow computation.

Note that i) an augmenting path in a transformed flow network can be also considered as a transformed flow network, and ii) the timestamp injection does not change the flow values of augmenting paths in a transformed flow network. Therefore, with no changes on $\mathsf{MF}[\tau'_s,\tau_e]$, we can conduct the timestamp injection for τ'_s on both N_f and the augmenting paths in P and P'. Let $P_{f'}$ denote the set of the reversed paths of the augmenting paths for a Maxflow f' from $\langle t,\tau_e\rangle$ to $\langle s',\tau_s\rangle$ found by `IncreMaxFlow⁻`. Then, as illustrated above, proving Lemma 5 is to prove that

$$N(P_{f'}) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right) = N(P) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right). \quad (7)$$

In the following, we show how this equation holds.

Step 2). With the assumption in Step i), we have an $\mathsf{MF}[\tau_s,\tau_e]$ (denoted by f) computed by finding the augmenting paths in P and P'. According to the insertion of edges in Lines 20-21 of Algorithm 3, we can know that the flow value of Maxflow f' from $\langle t,\tau_e\rangle$ to $\langle s',\tau_s\rangle$ equals to the sum of the capacities of the incoming edges of the virtual node $\langle s',\tau_s\rangle$, *i.e.*, $\sum_{p\in P} \mathsf{Flow}(p)$. The reason is that there must exist a Maxflow f'' from $\langle t,\tau_e\rangle$ to $\langle s',\tau_s\rangle$ having $|f''| = \sum_{p\in P} \mathsf{Flow}(p)$, which can always be obtained by using the reversed paths of the augmenting paths in P with the modification as follows: i) conduct timestamp injection on these reversed paths for τ'_s; and ii) for each reversed path, replace this path by its subpath p starting at $\langle t,\tau_e\rangle$ and ending at $\langle u,\tau'_s\rangle$, $u \ne s$ after inserting edge $\left(\langle u,\tau'_s\rangle, \langle s',\tau_s\rangle\right)$ into p. Therefore, we have $|f'| = |f''| = \sum_{p\in P} \mathsf{Flow}(p)$. Next, we show how Equation 7 holds by analyzing the relation between P_f and P.

For the set $P_{f'}$ consisting of the reversed paths of the augmenting paths for f' and the set P_Δ consisting of the reversed paths of the modified augmenting paths for f'', there are two exhausted cases.

- **Case 1:** $P_{f'} = P_\Delta$. For this case, we can easily deduce that $N(P_{f'}) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right) = N(P_\Delta) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right) = N(P) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$.
- **Case 2:** $P_{f'} \ne P_\Delta$. For this case, we modify the paths in $P_{f'}$ as follows: i) conduct timestamp injection on the augmenting paths in P for τ'_s; ii) replace each path in P by its subpath starting at $\langle s,\tau_s\rangle$ and ending at $\langle u,\tau'_s\rangle$, $u \ne s$; and iii) construct augmenting paths from $\langle s,\tau_s\rangle$ to $\langle t,\tau_e\rangle$ by combining paths in P and paths in $P_{f'}$ having the common nodes $\langle u,\tau'_s\rangle$, $u \ne s$. We replace the paths in $P_{f'}$ by these constructed augmenting paths. As the flow conservation holds for the nodes $\langle u,\tau'_s\rangle$, $u \ne s$, the paths in $P_{f'}$ together with the augmenting paths in P' can also find an $\mathsf{MF}[\tau_s,\tau_e]$ in $N_{[\tau_s,\tau_e]}$. Therefore, conducting $\left(\Delta_{\tau'_s}\left(N_f \uplus N(P_{f'})\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$ is equivalent to withdrawing flow value $\mathsf{Flow}(p)$ from the edges of each path p in $P_{f'}$ on the residual network N_{f_Δ} of $N_{[\tau_s,\tau_e]}$ *w.r.t.* Maxflow f_Δ. That is, $\left(\Delta_{\tau'_s}\left(N_f \uplus N(P_{f'})\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right) = \left(\Delta_{\tau'_s}\left(N_{f_\Delta} \uplus N(P_{f'})\right)\right) \setminus \left(N_{[\tau_s,\tau'_s]} \setminus N_{[\tau'_s,\tau'_s]}\right)$. Hence, Case 2 can be considered as Case 1 for the Maxflow f_Δ.

Putting all the analyses above together, Lemma 5 holds. □

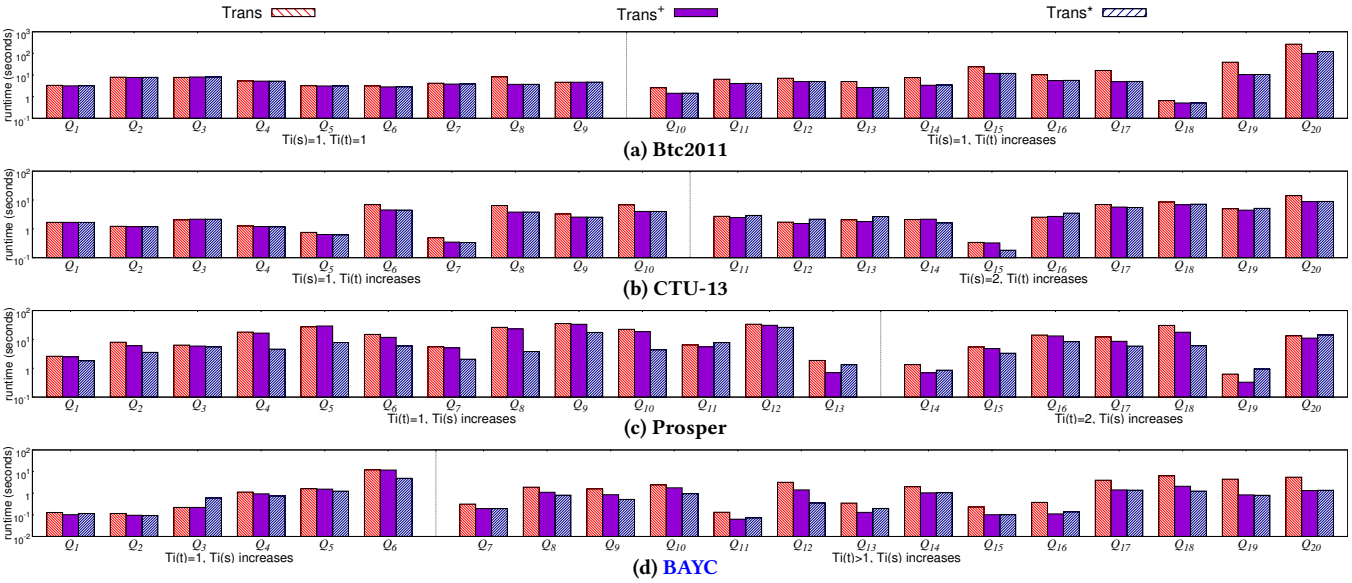


Figure 12: Runtimes of various network transformations on different temporal flow networks

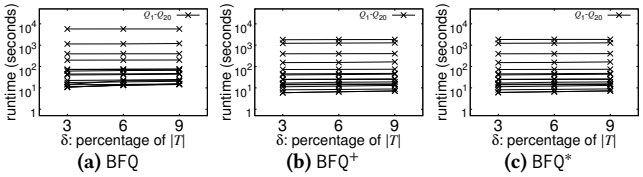


Figure 13: Runtimes of δ-BFlow solutions on the Btc2011 dataset when varying δ (EXP-5)

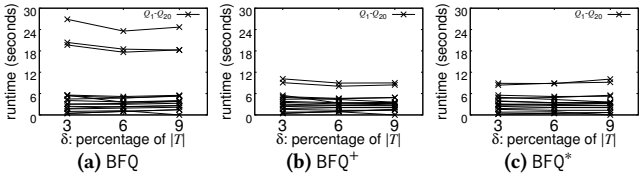


Figure 14: Runtimes of δ-BFlow solutions on the CTU-13 dataset when varying δ (EXP-5)

THEOREM 2. (**Correctness of** BFQ*) *Given a temporal flow network N_T with source s and sink t, and a parameter δ,* BFQ* *returns the flow density and the bursting interval of δ-*BFlow *from s to t.*

PROOF. Similar to the proof of Theorem 1, we prove Theorem 2 by showing that BFQ* also computes Maxflows *w.r.t.* the same candidate intervals enumerated in BFQ.

Let τ_{min} denote the timestamp $\min_{\tau \in \mathsf{Ti}(s)}\{\tau\}$. Then, for the candidate interval $[\tau_s, \tau_s + \delta]$, $\tau_s \in \mathsf{Ti}(s)$ enumerated in Lines 3-4 of BFQ, BFQ* computes $\mathsf{MF}[\tau_{min}, \tau_{min} + \delta]$ and $\mathsf{MF}[\tau_s, \tau_s + \delta]$, $\tau_s \in \mathsf{Ti}(s) - \{\tau_{min}\}$ by using Dinic in Lines 4-5 and the incremental procedure `IncreMaxFlow⁻` in Lines 6-13, respectively. The correctness of `IncreMaxFlow⁻` is guaranteed by Lemmas 4-5. For the rest candidate intervals $[\tau_s, \tau_e]$, $\tau_s \in \mathsf{Ti}(s)$, $\tau_e \in \mathsf{Ti}(t)$ enumerated in Lines 3-4 of BFQ, BFQ* computes Maxflows *w.r.t.* these intervals by using the incremental procedure `IncreMaxFlow⁺` in Line 14. Therefore, BFQ* and BFQ compute Maxflows *w.r.t.* the same candidate intervals, and hence, Theorem 2 holds. □

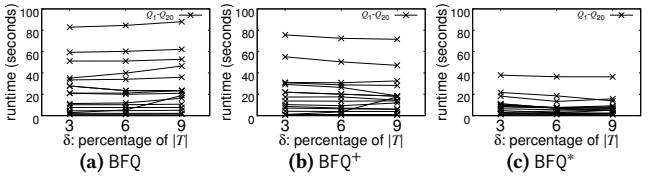


Figure 15: Runtimes of δ-BFlow solutions on Prosper when varying δ (EXP-5)

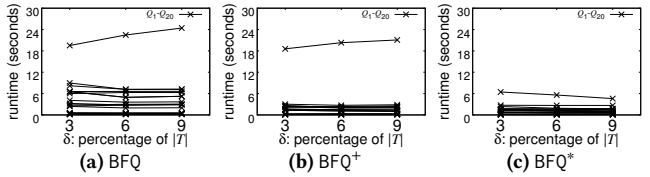


Figure 16: Runtimes of δ-BFlow solutions on BAYC when varying δ (EXP-5)

C Experimental Results

This section shows some detailed experimental results of Section 6.

EXP-4. Runtimes of the network transformations for the proposed δ-BFlow solutions. Let `Trans`, `Trans⁺`, and `Trans*` denote all the network transformation (from scratch or incrementally) of BFQ, BFQ⁺ and BFQ*, respectively. Figure 12 shows the runtimes of `Trans`, `Trans⁺`, and `Trans⁺⁻` on Btc2011, CTU-13, Prosper, and BAYC, respectively. It can be observed from Figure 12 that, when compared to `Trans`, the runtimes of `Trans⁺` and `Trans*` had similar trends of speedup as that of the results shown in Figure 9.

EXP-5. Runtimes of δ-BFlow solutions when varying δ. The runtimes of our proposed δ-BFlow solutions on the four used datasets when varying δ are presented in Figures 13-16, respectively. We obtained similar observations (as shown in EXP-5 in Section 6.2) from the experimental results on these four datasets.