*Article*

# Approximating the Temporal Neighbourhood Function of Large Temporal Graphs

**Pierluigi Crescenzi [1,†], Clémence Magnien [3] and Andrea Marino [4*]**

[1] Université de Paris, IRIF, CNRS, F-75013 Paris, France
[3] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[4] Dipartimento di Statistica, Informatica, Applicazioni, Università degli Studi di Firenze, Viale Morgagni, 59, 50134 Firenze, Italy
* Correspondence: andrea.marino@unifi.it
† On-leave from Università degli Studi di Firenze, DiMaI, I-50134 Firenze, Italy

Received: date; Accepted: date; Published: date

**Abstract:** Temporal networks are graphs in which edges have temporal labels, specifying their starting times and their traversal times. Several notions of distances between two nodes in a temporal network can be analyzed, by referring, for example, to the earliest arrival time or to the latest starting time of a temporal path connecting the two nodes. In this paper we mostly refer to the notion of temporal reachability by using the earliest arrival time. In particular, we first show how the sketch approach, which has been already used in the case of classical graphs, can be applied to the case of temporal networks in order to approximately compute the sizes of the temporal cones of a temporal network. By making use of this approach, we subsequently show how we can approximate the temporal neighborhood function (that is, the number of pairs of nodes reachable from one another in a given time interval) of large temporal networks in a few seconds. Finally, we apply our algorithm in order to analyze and compare the behavior of 25 public transportation temporal networks. Our results can be easily adapted to the case in which we want to refer to the notion of distance based on the latest starting time.

**Keywords:** temporal network, link stream, temporal path, earliest arrival time, temporal reachability, neighborhood function, public transportation system

## 1. Introduction

Temporal networks are graphs in which nodes and edges can appear or disappear over time, due not only to failures or malfunctioning of the entities participating to the system represented by the temporal graph, but mostly to the "normal" behaviour of the system itself. A typical temporal network is a person-to-person communication network within a company. In such a network, for example, nodes can appear or disappear (depending on the recruitment policy of the company), and edges appear whenever an employee of the company sends an e-mail message to another employee of the company. In this paper, we will focus on temporal networks in which the set of nodes does not change over time (at least over a specified interval of time). Moreover, we consider only the case in which edges are available at discrete time instants, so that the dynamics of the network is specified only by the appearance times of the edges.

As observed in [1], temporal networks can model a great variety of systems in nature, society and technology. Several different types of temporal networks have, indeed, been analyzed: person-to-person communication networks (such as e-mails or phone calls), one-to-many information spreading networks (such as Twitter interactions), contact networks (such as cell phone proximity detections), biological networks (such as protein interactions), distributed computing networks (such as autonomous system communications), infrastructure networks (such as public transport timetables), and many others. It is worth observing, however, that different names have been used for denoting temporal networks (even though the basic notion was almost the same), such as, for example, dynamic networks [2], time-varying graphs [3], evolving networks [4], and link streams [5].

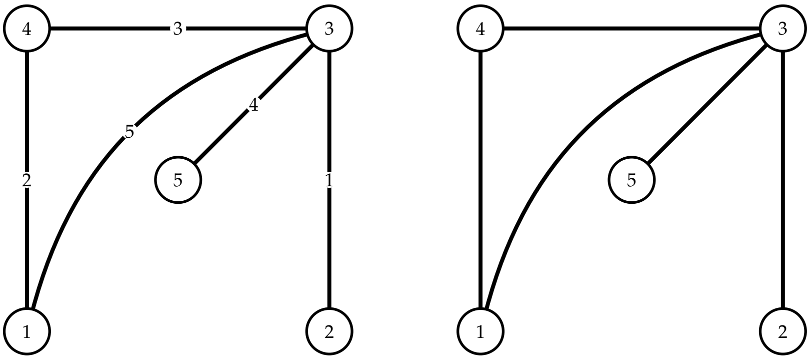

**Figure 1.** An example of temporal graph with 5 nodes and 5 temporal edges (left), and the corresponding "non-temporal" graph in which edge temporal labels have been removed (right).

In this paper we are interested in studying reachability properties of temporal networks. As already observed in [6], if we just remove all time information from the temporal graph (and collapse, if necessary, multiple edges between any two vertices into a single edge), we clearly loose all the temporal information of the graph. This loss can be critical to the understanding of the reachability relationships between the nodes of the graph. For example, in the left part of Figure 1 a temporal network with five nodes and five temporal edges is represented, while in the right part of the figure the "non-temporal" version of the graph (in which all edge temporal labels have been removed) is shown. It is easy to verify that the two simple paths from node 1 to node 2 in the non-temporal graph do not exist in the temporal network (a temporal path is, intuitively, a path such that each edge in the path appears later than the edges preceding it in the path). Indeed, the edge from node 3 to node 2 appears at time 1, hence it cannot be used within a path starting from node 1, since all this node's edges appear after that time. Moreover, the path of length 2 from node 1 to node 5 in the non-temporal graph does not exist in the temporal graph since it is only possible to reach node 3 from node 1 in one step at time 5, while the edge from node 3 to node 5 appears at time 4. In summary, removing temporal information may let us conclude that two nodes (i.e. 1 and 2) are reachable, while they are not, or that the length of the shortest path between two nodes (i.e. 1 and 5) is smaller than it really is.

For this reason, we are interested in developing algorithmic techniques which allow us to efficiently compute aggregate information about temporal paths and time-distances between pairs of nodes. In particular, we focus on the *temporal neighborhood function* (in short, TNF) of a temporal network, which is the natural extension of the neighborhood function already widely analyzed in the case of non-temporal graphs [7,8]. More precisely, given a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, the temporal neighborhood function returns the value $|\mathcal{N}(\mathbb{I})|$, where $\mathcal{N}(\mathbb{I})$ denotes the set of pairs of nodes $(u, v)$ such that there exists a *temporal path* from $u$ to $v$, which starts from $u$ not earlier than $t_\alpha$, arrives in $v$ no later than $t_\omega$, and it is such that each edge appears later than the edges preceding it.

By assuming that a temporal network is represented by the sequence of its temporal edges, ordered in non-decreasing order with respect to their appearance times, the temporal neighborhood function can be easily computed by making use of the following "scan-based" algorithm [9], which allows us to compute the cardinality of the *temporal cone* of a node $s$, which is the set of nodes reachable from $s$ in the interval $[t_\alpha, t_\omega]$.

```
1. Initialize $t[s] = t_\alpha$ and $t[v] = \infty$ for all $v \neq s$.
2. For each (directed) edge $e = (u, v, t)$:
    (a) If $t \leq t_\omega$, $t[u] < t$, and $t + 1 < t[v]$, then $t[v] = t + 1$.
    (b) If $t > t_\omega$ then stop.
3. Return the number of elements of $t$ different from $\infty$.
```

If $m$ is the number of temporal edges, it is easy to verify that the complexity of the above algorithm is $O(m)$. In order to compute the temporal neighborhood function, we have to execute the above procedure starting from every node of the temporal network. We refer to this algorithm as ETNF (Exact TNF). Hence, if $n$ denotes the number of nodes, we can conclude that ETNF runs in time $O(nm)$.

Unfortunately, this time complexity is not acceptable when dealing with real-world temporal networks, where there are millions of nodes and billions of temporal edges. For this reason, in this paper we propose a new algorithm called ATNF for approximating the temporal neighborhood function of a temporal network.

### 1.1. Our results

In order to approximate the temporal neighborhood function, we first describe a simple dynamic programming algorithm for computing the *reverse temporal cone* of a node $s$, which is the set of nodes that can reach $s$ in a specific interval $[t_\alpha, t_\omega]$. Note that, if we can compute the cardinalities of the reverse temporal cones, then we can also compute the temporal neighborhood function. We then show how the *sketch* approach, which has been already widely used in the case of non-temporal graphs [10–13], can be applied to the case of temporal networks in order to approximately compute the cardinalities of the reverse temporal cones of a temporal network. More specifically, the resulting approximation algorithm, called ATNF (Approximated TNF), has relative error bounded by $\epsilon$ with high probability, whenever the sketches have size $k = \Theta\left(\frac{\log n}{\epsilon^2}\right)$. The time complexity of the algorithm is $O(km)$.

We then experimentally evaluate the quality of the approximation performed by ATNF by comparing the approximate value of the temporal neighborhood function with the exact one (computed by making use of the scan-based exact algorithm described in the previous section) on a data-set containing several medium-size temporal networks. As a matter of fact (and as expected), the obtained approximation is much better than the one guaranteed in theory, even when the size of the sketches is significantly smaller than the required size. By making use of the approximation algorithm, we hence show how we can accurately approximate, in a few minutes, the temporal neighborhood function (and, hence, the distance distribution) of two large temporal networks: the IMDB collaboration network (which is undirected) and the Twitter re-tweets network (which is directed). The first network contains more than half a million nodes and more than three millions edges, while the second network contains more than two millions nodes and more than sixteen millions edges.

Finally, we apply ATNF in order to analyze and compare the behavior of twenty-five public transportation temporal networks [14]. In particular, we analyze the reachability properties of these networks, by computing the values of the temporal neighborhood function in different intervals during a day. As a matter of fact, we observe that there are cities which perform significantly better than others with respect to these reachability properties, and that this result cannot always been deduced by simply looking at the density of the temporal network. Moreover, we show that the quality of the approximation is preserved even with small values of the sketch sizes: this allowed us to perform the entire experimental evaluation for all the cities in less than one hour (while the exact algorithm would have roughly required almost four days).

### 1.2. Related work

Algorithms for distance computation in temporal networks have been proposed in [6] and [15]. [16] uses a similar algorithm to [6] for computing spanning trees. [9] proposes a method for indexing data and answer reachability and path queries in temporal networks. The problem of finding different types of path (minimum traveling time, earliest arrival time, and minimum number of hops) in temporal network has also been investigated in a distributed context [17].

A problem related to the one analyzed in this paper is computing distances in road networks, where edges have a traversal time but are supposed to exist at all times. In this case, the focus is mainly on pre-computing some information in order to be able to answer fastest path queries very quickly (see, for example, [18,19]). [20] focuses on the impact of the passenger demand on the performance of path finding algorithms.

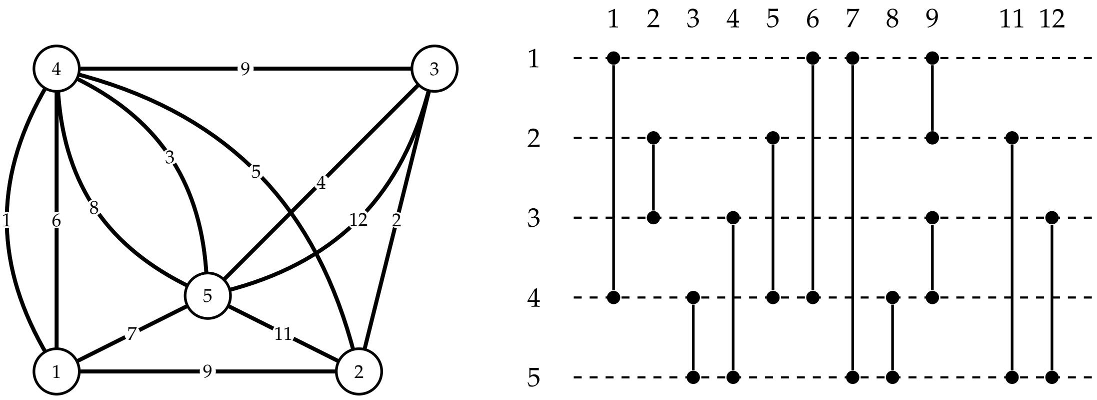

**Figure 2.** An example of temporal graph with 5 nodes and 12 temporal edges (labels on edges represent appearing time), and the corresponding link stream representation (labels on the left represent nodes, labels above represent time instants).

Concerning public transportation networks, many works have focused on the efficient *design* of such networks, by considering, e.g., where to place hubs, see for instance [21,22], or on their *robustness* or *resilience*, by studying how perturbations on specific nodes or links, or changes in demand, affect the whole network, see for instance [23,24]. Other works have used connectivity notions to evaluate the importance of nodes in transportation networks, see for instance [25] and references within.

### 1.3. Structure of the paper

In Section 2, we give all basic definitions and notations concerning temporal networks, paths, cones and neighborhood functions. In Section 3, we describe the dynamic programming algorithm for computing reverse temporal cones, and we show how the bottom-$k$ sketch approach can be used in order to approximate the cardinalities of these cones (and, hence, the temporal neighborhood function), getting our approximation algorithm ATNF. In Section 4 we experimentally evaluate the quality of the approximation of ATNF and its running time, comparing our results with the ones of ETNF. Moreover, we use the algorithm itself to compute the temporal neighborhood functions of two large temporal networks. Finally, in Section 5, we apply ATNF for comparing the reachability properties of twenty-five public transportation networks.

## 2. Definitions and notations

The following definitions are mostly inspired by [3,5,9]. A *temporal graph* is a pair $G = (V, E)$, where $V$ is the set of *nodes* and $E$ is the set of *temporal edges*. A temporal edge $e \in E$ is a triple $(u, v, t)$, where $u, v \in V$ are the source and destination nodes of the edge, respectively, and $t \in \mathbb{N}$ is its *appearing time*. When the temporal edges are bidirectional, then $(u, v, t)$ can be also written as $(v, u, t)$. The *time horizon* $\mathcal{T}(G)$ of a temporal graph $G$ is the union of all the appearing times of its temporal edges. For instance, in the left part of Figure 2 a temporal graph with 5 nodes and 12 temporal edges is shown: its time horizon is $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12\}$. A different way of representing a temporal graph is the *link stream* diagram shown in the right part of Figure 2, where the labels on the left of the diagram represent the nodes of the graph while the labels above the diagram represent the different time instants. In this paper, we will indeed assume that the edges are given to the algorithms one after the other (similarly to the streaming model) in non-decreasing order with respect to the appearing time: this corresponds to reading the edges in the link stream diagram from left-to right.

### 2.1. Temporal paths

A *temporal path* $\mathbb{P}$ in a temporal graph $G = (V, E)$ from a node $u \in V$ to a node $v \in V$ is a sequence of temporal edges $e_1 = (u_1, v_1, t_1), e_2 = (u_2, v_2, t_2), \ldots, e_k = (u_k, v_k, t_k)$ such that $u = u_1$,

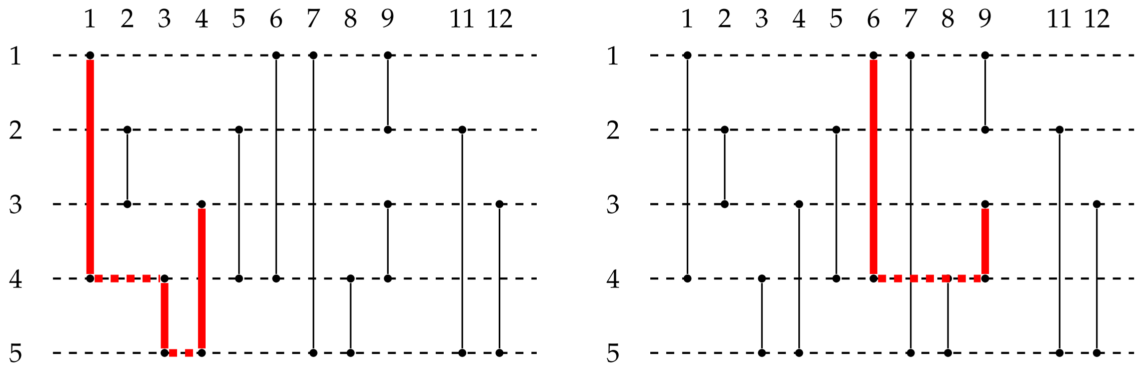

**Figure 3.** The earliest arrival path from node 1 to node 3 in the temporal graph of Figure 2 in the time interval $[1, 12]$ (left) and in the time interval $[2, 12]$ (right).

$v = v_k$, and, for each $i$ with $1 < i \leq k$, $u_i = v_{i-1}$ and $t_i \geq t_{i-1} + 1$. The *length* of a temporal path is the number of temporal edges it contains. For example, referring to the temporal graph of Figure 2, $(1, 4, 1), (4, 2, 5), (2, 5, 11), (5, 3, 12)$ is a temporal path of length 4. On the contrary, $(1, 4, 1), (4, 2, 5), (2, 3, 2)$ is not a temporal path, since the appearing time of the second edge is larger than the appearing time of the third edge. The *starting time* (respectively, *ending time*) of a temporal path $\mathbb{P}$, denoted by $\sigma(\mathbb{P})$ (respectively, $\eta(\mathbb{P})$), is equal to the appearing time of the first (respectively, last) edge in the path. For instance, if $\mathbb{P} = (1, 4, 1), (4, 2, 5), (2, 5, 11), (5, 3, 12)$, then $\sigma(\mathbb{P}) = 1$ and $\eta(\mathbb{P}) = 12$. Given a time interval $\mathbb{I} = [t_\alpha, t_\omega]$ and two nodes $u$ and $v$, we will denote by $\mathcal{P}(u, v, \mathbb{I})$ the set of all temporal paths $\mathbb{P}$ from $u$ to $v$ such that $t_\alpha \leq \sigma(\mathbb{P}) \leq \eta(\mathbb{P}) \leq t_\omega$. Among all temporal paths between two nodes in a given time interval, in this paper we will distinguish the ones which allows us to arrive as early as possible.

**Definition 1.** *Given a temporal graph $G = (V, E)$, two nodes $u$ and $v$ in $V$, and a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, a path $\mathbb{P} \in \mathcal{P}(u, v, \mathbb{I})$ is said to be an **earliest arrival path** if $\eta(\mathbb{P}) = \min\{\eta(\mathbb{P}') : \mathbb{P}' \in \mathcal{P}(u, v, \mathbb{I})\}$.*

For example, the left part of Figure 3 shows the earliest arrival path from node 1 to node 3 in the time interval $[1, 12]$ in the temporal graph of Figure 2: this path consists of the temporal edges $(1, 4, 1)$, $(4, 5, 3)$, and $(5, 3, 4)$, its starting time is 1, and its ending time is 4. The right part of the figure, instead, shows an earliest arrival path from node 1 to node 3 in the time interval $[2, 12]$: it consists of the temporal edges $(1, 4, 6)$ and $(4, 3, 9)$, its starting time is 6, and its ending time is 9 (note that another earliest arrival path from node 1 to node 3 in the same interval is the one consisting of the temporal edges $(1, 5, 7)$, $(5, 4, 8)$, and $(4, 3, 9)$).

### 2.2. Temporal reachability cones and neighborhood functions

Given a temporal graph $G = (V, E)$, a node $u \in V$, and a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, in this paper we are interested in efficiently computing the number of pairs of nodes $(u, v)$ such that $v$ can be reached from $u$ in the interval $\mathbb{I}$. To this aim, we give the following definition.

**Definition 2.** *Given a temporal graph $G = (V, E)$, a node $u$, and a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, the **temporal reachability cone** of $u$ in the interval $\mathbb{I}$ is defined as $\mathcal{C}(u, \mathbb{I}) = \{u\} \cup \{v \in V : \mathcal{P}(u, v, \mathbb{I}) \neq \emptyset\}$.*

For example, by referring to the temporal graph of Figure 2, the left part of Figure 4 shows the temporal reachability cone of node 1 in the interval $[1, 5]$: in this case, all nodes can be reached by node 1. The right part of figure, instead, shows the temporal reachability cone of node 1 in the interval $[9, 12]$: in this case, all nodes can be reached by node 1 apart from node 4. It is also easy to verify that the temporal reachability cone of node 1 in the interval $[2, 5]$ contains only node 1, since there are no temporal edges leaving it in this time interval.

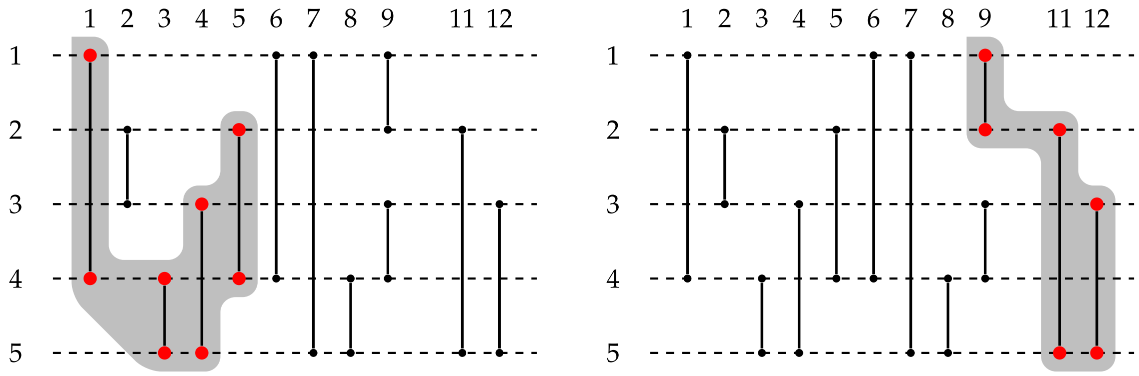

**Figure 4.** Two temporal reachability cones of node 1: the first one (left) is in the interval $[1,5]$ and includes all nodes, while the second one (right) is in the interval $[9,12]$ and includes all nodes but node 4.

**Definition 3.** *Given a temporal graph $G = (V, E)$ and a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, the* ***temporal neighborhood function*** *in the interval $\mathbb{I}$ is defined as $|\mathcal{N}(\mathbb{I})|$, where $\mathcal{N}(\mathbb{I}) = \{(u, v) \in V \times V : v \in \mathcal{C}(u, \mathbb{I})\}$.*

For example, by referring to the temporal graph of Figure 2 and by choosing $\mathbb{I} = [1,5]$, $\mathcal{N}(\mathbb{I})$ contains the pairs $(1,1)$, $(1,2)$, $(1,3)$, $(1,4)$, and $(1,5)$, since all nodes belong to $\mathcal{C}(1,\mathbb{I})$. $\mathcal{N}(\mathbb{I})$ also contains the pairs $(2,2)$, $(2,3)$, $(2,4)$, $(2,5)$ (since $\mathcal{C}(2,\mathbb{I}) = \{2,3,4,5\}$), $(3,2)$, $(3,3)$, $(3,4)$, $(3,5)$ (since $\mathcal{C}(3,\mathbb{I}) = \{2,3,4,5\}$), $(4,1)$, $(4,2)$, $(4,3)$, $(4,4)$, $(4,5)$ (since $\mathcal{C}(4,\mathbb{I}) = \{1,2,3,4,5\}$), $(5,2)$, $(5,3)$, $(5,4)$, and $(5,5)$ (since $\mathcal{C}(5,\mathbb{I}) = \{2,3,4,5\}$). Therefore the temporal neighborhood function in $\mathbb{I}$ is equal to 22.

## 3. Computing temporal neighborhood functions through reverse temporal cones

As we already said, in this paper we consider the *left-to-right* order for reading the stream of temporal edges, which corresponds to reading the edges in non-decreasing order with respect to their appearing times.

**Definition 4.** *Given a temporal graph $G = (V, E)$ and a time instant $t \in \mathcal{T}(G)$, the* ***predecessor*** $\mathtt{pred}(t)$ *of $t$ in $G$ is the maximum time instant $t' \in \mathcal{T}(G)$ such that $t' < t$ (if $t'$ does not exist, then we set $\mathtt{pred}(t) = \bot$).*

For example, by referring to the temporal graph of Figure 2, $\mathtt{pred}(1) = \bot$, $\mathtt{pred}(9) = 8$, and $\mathtt{pred}(11) = 9$.

Observe that, if $\mathbb{I} = [t_\alpha, t_\omega]$ and $\mathtt{pred}(t_\omega) \neq \bot$, then $\mathcal{N}(\mathbb{I}) - \mathcal{N}([t_\alpha, \mathtt{pred}(t_\omega)])$ contains exactly all pairs of nodes $(u, v)$ such that any earliest arrival path $\mathbb{P}$ from $u$ to $v$ satisfies $\eta(\mathbb{P}) = t_\omega$.

In order to compute the temporal neighborhood function of a temporal graph $G$ in a given time interval $[t_\alpha, t_\omega]$, we first introduce the following definition.

**Definition 5.** *Given a temporal graph $G = (V, E)$, a node $u$, and a time interval $\mathbb{I} = [t_\alpha, t_\omega]$, the* ***reverse temporal cone*** *of $u$ in the interval $\mathbb{I}$ is defined as $\mathcal{R}(u, \mathbb{I}) = \{v \in V : \mathcal{P}(v, u, \mathbb{I}) \neq \emptyset\}$.*

In other words, the reverse temporal cone of $u$ in the interval $\mathbb{I}$ contains all nodes $v$ that can reach $u$ in $\mathbb{I}$ (hence, if $v \in \mathcal{R}(u, \mathbb{I})$, then $u \in \mathcal{C}(v, \mathbb{I})$). The advantage of referring to reverse temporal cones is that these cones can be easily computed by using the following dynamic programming algorithm (observe that, for any $t \in \mathcal{T}(G)$, $\mathcal{C}(u, [t,t])$ is the neighborhood of $u$ at time $t$, that is, the set of nodes $v$ such that $(u, v, t) \in E$).

**Base case** $\mathcal{R}(u, [t_\alpha, t_\alpha]) = \{u\} \cup \{v \in V : u \in \mathcal{C}(v, [t_\alpha, t_\alpha])\}$.
**Recursive step** Let $t \in \mathcal{T}(G)$ with $t > t_\alpha$ and suppose we have computed $\mathcal{R}(u, [t_\alpha, t'])$ for every $t' < t$ with $t' \in \mathcal{T}(G)$. Then, $\mathcal{R}(u, [t_\alpha, t]) = \mathcal{R}(u, [t_\alpha, \mathtt{pred}(t)]) \cup \bigcup_{v : u \in \mathcal{C}(v, [t,t])} \mathcal{R}(v, [t_\alpha, \mathtt{pred}(t)])$.

The base case simply states that, if there is an edge $(v, u, t_\alpha)$, then $v$ belongs to the reverse temporal cone of $u$ in the interval $[t_\alpha, t_\alpha]$. The recursive step, instead, says that, if there is an edge $(v, u, t)$, then all nodes that could reach $v$ before $t$ can now also reach $u$ at time $t$. For example, by referring to the temporal graph of Figure 2 (in which all edges are bidirectional), the following table shows the evolution of the reverse temporal cones $\mathcal{R}(u, [1, 6])$ according to the above algorithm (until all nodes are reachable from all other nodes).

| $(u, v, t)$ | $\mathcal{R}(1, [1, 6])$ | $\mathcal{R}(2, [1, 6])$ | $\mathcal{R}(3, [1, 6])$ | $\mathcal{R}(4, [1, 6])$ | $\mathcal{R}(5, [1, 6])$ |
|---|---|---|---|---|---|
| $(1, 4, 1)$ | $\{1, 4\}$ | $\{2\}$ | $\{3\}$ | $\{1, 4\}$ | $\{5\}$ |
| $(2, 3, 2)$ | $\{1, 4\}$ | $\{2, 3\}$ | $\{2, 3\}$ | $\{1, 4\}$ | $\{5\}$ |
| $(4, 5, 3)$ | $\{1, 4\}$ | $\{2, 3\}$ | $\{2, 3\}$ | $\{1, 4, 5\}$ | $\{1, 4, 5\}$ |
| $(3, 5, 4)$ | $\{1, 4\}$ | $\{2, 3\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ |
| $(2, 4, 5)$ | $\{1, 4\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ |
| $(1, 4, 6)$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ | $\{1, 2, 3, 4, 5\}$ |

Let us assume that the set of temporal edges of a temporal graph $G = (V, E)$ is ordered in non-decreasing order with respect to the appearing times, and that edges are not bidirectional. Let $\overrightarrow{E} = (e_{\overrightarrow{1}}, e_{\overrightarrow{2}}, \ldots, e_{\overrightarrow{m}})$ be the resulting ordered set of temporal edges, so that the appearing time of $e_{\overrightarrow{i}}$ is not greater than the appearing time of $e_{\overrightarrow{i+1}}$, for any $i$ with $1 \le i < m$. We can then implement the above dynamic programming algorithm by scanning the edges in $\overrightarrow{E}$ one after the other. For each edge $e_{\overrightarrow{i}} = (v, u, t)$, if $t = t_\alpha$, then we add $v$ to $\mathcal{R}(u, [t_\alpha, t_\alpha])$. Otherwise (that is, $t > t_\alpha$), we set $\mathcal{R}(u, [t_\alpha, t])$ equal to the union of $\mathcal{R}(u, [t_\alpha, t])$ and $\mathcal{R}(v, [t_\alpha, \texttt{pred}(t)])$, after having initialized $\mathcal{R}(u, [t_\alpha, t])$ to $\mathcal{R}(u, [t_\alpha, \texttt{pred}(t)])$ the first time an edge appearing at time $t$ is scanned. Since the size of the intermediate reverse temporal cones can be linear in the number $n$ of nodes in the graph, the time complexity of this algorithm is $O(nm)$. Moreover, if we want to maintain all the intermediate reverse temporal cones, then the space complexity of the algorithm is $O(n^2 m)$. However, if we just want to compute the final reverse temporal cones, then the space complexity can be reduced to $O(n^2)$. Observe that, if the edges are bidirectional, then we can easily modify this algorithm by simply considering twice each edge $(v, u, t)$: once as $(v, u, t)$, and the other as $(u, v, t)$.

Once we have computed the reverse temporal cones in a specific interval $\mathbb{I}$, we can easily compute $\mathcal{N}(\mathbb{I})$. Indeed, we have that, for each pair of nodes $u$ and $v$, $(u, v)$ belongs to $\mathcal{N}(\mathbb{I})$ if and only if $u \in \mathcal{R}(v, \mathbb{I})$. This implies that

$$|\mathcal{N}(\mathbb{I})| = \sum_{u \in V} |\mathcal{R}(u, \mathbb{I})| .$$

Hence, the temporal neighborhood function can be computed by using the sizes of the reverse temporal cones.

**Remark.** Note that the above dynamic programming approach does not seem to be applicable to the computation of temporal cones, as defined in the previous section. More precisely, it is not clear how to rewrite the recursive step of the algorithm, when referring to temporal cones. Indeed, in general it holds that

$$\mathcal{C}(u, [t_\alpha, t]) \ne \mathcal{C}(u, [t_\alpha, \texttt{pred}(t)]) \cup \bigcup_{v \in \mathcal{C}(u, [t,t])} \mathcal{C}(v, [t_\alpha, \texttt{pred}(t)]),$$

since, for each vertex $v$, $\mathcal{C}(v, [t_\alpha, \texttt{pred}(t)])$ is the set of all the vertices reachable from $v$ through temporal paths which use edge appearing times smaller than $t$. Hence, $\mathcal{C}(u, [t_\alpha, t])$ does not necessarily include the nodes in $\mathcal{C}(v, [t_\alpha, \texttt{pred}(t)])$, as the corresponding paths might turn out to be not valid from a temporal point of view. Indeed, the edge from $u$ to $v$ appears at time $t$ but the edges on the paths from $v$ to other vertices may have appearing times smaller than $t$. Let us consider, for example, the temporal graph with the three nodes 1, 2, and 3, and the two temporal edges $(1, 2, 1)$ and $(2, 3, 2)$. Clearly, $\mathcal{C}(2, [1, 1]) = \{1, 2\}$ and $\mathcal{C}(3, [1, 1]) = \{3\}$. Hence, when the edge $(2, 3, 2)$ is analyzed, we have that

$$\mathcal{C}(3, [1, 2]) = \{2, 3\} \ne \{1, 2, 3\} = \mathcal{C}(3, [1, 1]) \cup \mathcal{C}(2, [1, 1]).$$

### *3.1. Approximating the size of the reverse temporal cones*

As we have already observed, the time complexity of the dynamic programming algorithm for computing the reverse temporal cones is $O(nm)$, where $n = |V|$ and $m = |E|$. This complexity can turn out to be prohibitive when dealing with temporal graphs with thousands of nodes and millions of temporal edges. For this reason, in this section we propose ATNF, an algorithm for computing an approximation of the size of the reverse temporal cones, which is based on the application of the *sketch techniques*: these techniques allow us to represent the reverse temporal cone for each node $u$ in a compressed approximated form of (almost) constant size $k$ (typically $O(\log(n))$). By using cone sketches in place of real cones, we will obtain a dynamic programming approximation algorithm whose time complexity is $O(km)$ time.

#### 3.1.1. Sketch operations

Given a set $A$, a sketch $S(A)$ is a compressed form of representation of $A$ of size $O(k)$, with $k \in \mathbb{N}$, providing the following operations.

INIT $(S(A))$ How a sketch $S(A)$ for $A$ is initialized.
UPDATE $(S(A), u)$ How a sketch $S(A)$ for $A$ is modified when an element $u$ is added to $A$.
UNION $(S(A), S(B))$ Given two sketches for $A$ and $B$, provide a sketch for $A \cup B$.
SIZE $(S(A))$ Estimate the number of distinct elements of $A$.

The following two requirements are needed for sketches: (i) given two sketches $S(A)$ and $S(B)$ for any two sets $A$ and $B$, $S(A \cup B)$ can be computed just by looking at $S(A)$ and $S(B)$, and (ii) the order in which the elements are added and adding any element twice does not affect the sketch. We assume that $|A| > k > 1$.

#### 3.1.2. Bottom-$k$ sketches

One of the most popular sketch techniques is the *bottom-k* technique, which works as follows. Given a mapping $r : U \rightarrow \{1/n, 2/n, \ldots, n/n\}$ and a subset $A$ of $U$, we denote as $H_k(A)$ the first $k$ elements of $A$ according to $r$ (or $A$ itself if $|A| < k$). A *bottom-k sketch* for $A$ is $H_k(A)$.

INIT $(S(A))$: $S(A) = \emptyset$.
UPDATE $(S(A), u)$: return $H_k(S(A) \cup \{u\})$.
UNION $(S(A), S(B))$: return $H_k(S(A) \cup S(B))$.
SIZE $(S(A))$: If $|S(A)| < k$ return $|S(A)|$. Otherwise return $\frac{k-1}{\max_{u \in S(A)} r(u)}$.

As an example, let $U$ be the set of the 26 letters of the alphabet and let $r : U \rightarrow \{1/26, 2/26, \ldots, 26/26\}$ be the following function:

| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ | $j$ | $k$ | $l$ | $m$ | $n$ | $o$ | $p$ | $q$ | $r$ | $s$ | $t$ | $u$ | $v$ | $w$ | $x$ | $y$ | $z$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $r$ | $\frac{10}{26}$ | $\frac{22}{26}$ | $\frac{2}{26}$ | $\frac{24}{26}$ | $\frac{25}{26}$ | $\frac{12}{26}$ | $\frac{6}{26}$ | $\frac{7}{26}$ | $\frac{11}{26}$ | $\frac{17}{26}$ | $\frac{13}{26}$ | $\frac{15}{26}$ | $\frac{14}{26}$ | $\frac{19}{26}$ | $\frac{5}{26}$ | $\frac{9}{26}$ | $\frac{21}{26}$ | $\frac{8}{26}$ | $\frac{18}{26}$ | $\frac{26}{26}$ | $\frac{16}{26}$ | $\frac{1}{26}$ | $\frac{20}{26}$ | $\frac{4}{26}$ | $\frac{23}{26}$ | $\frac{3}{26}$ |

Let $A = \{a, l, i, c, e\}$, $B = \{w, o, n, d, e, r, l, a\}$, and $k = 3$. Then $S(A)$ is $\{a, i, c\}$ and $S(B)$ is $\{o, r, a\}$, as these letters are the three elements of $A$ and $B$, respectively, having minimum $r$-values. Hence,

$$S(A \cup B) = \texttt{UNION}(S(A), S(B)) = H_3(S(A) \cup S(B)) = H_3(\{a, i, c, o, r\}) = \{c, o, r\}.$$

The size of $A \cup B = \{a, l, i, c, e, w, o, n, d, r\}$, which is equal to 10, can then be estimated as follows:

$$\texttt{SIZE}(S(A \cup B)) = \texttt{SIZE}(\{c, o, r\}) = \frac{2}{\max_{u \in \{c,o,r\}} r(u)} = \frac{2}{8/26} = \frac{52}{8} = 6.5.$$

| **Undirected graphs** | | | **Directed graphs** | | |
|---|---|---|---|---|---|
| *Name* | *Nodes* | *Edges* | *Name* | *Nodes* | *Edges* |
| TOPOLOGY | 34 761 | 154 842 | ENRON | 150 | 24 705 |
| ROLLERNET | 63 | 403 834 | COLLEGE | 1 900 | 59 834 |
| IMDB- | | | WIKI-ELECTION | 7 118 | 103 675 |
| ADVENTURE | 47 763 | 157 492 | FACEBOOK-WALLPOST | 46 952 | 876 993 |
| HORROR | 65 338 | 167 026 | TWITTER | 3 511 241 | 16 438 790 |
| THRILLER | 69 753 | 188 862 | | | |
| CRIME | 62 050 | 216 741 | | | |
| ROMANCE | 79 227 | 305 390 | | | |
| ACTION | 72 260 | 338 815 | | | |
| COMEDY | 162 303 | 666 568 | | | |
| DRAMA | 279 059 | 1 342 886 | | | |
| ALL | 527 535 | 3 152 994 | | | |

**Table 1.** Number of nodes and edges of each (undirected or directed) graph.

It can be shown that the mean relative error of the sketch sizes with respect to the real sizes is bounded by $0.79/\sqrt{(k-2)}$ and that if we choose $k \in \Theta\left(\frac{\log |U|}{\epsilon^2}\right)$, the mean relative error is bounded by $\epsilon$ with high probability [10,11].

### 3.1.3. Applying sketches to reverse temporal cones

In the case of reverse temporal cones, the sets whose sizes we want to approximate are the reverse temporal cones at different time instants within a specific time interval $\mathbb{I}$. Indeed, for each edge scanned by the dynamic programming algorithm described in the previous section, either we use the `UPDATE` operation in order to add a node to the sketch of a reverse temporal cone, or we use the `UNION` operation in order to compute the union of the sketches of two reverse temporal cones. Whenever all edges whose appearing time is in $\mathbb{I}$ have been read, we can use the `SIZE` operation in order to estimate, for each node $u$, the size of $\mathcal{R}(u, \mathbb{I})$. Let $\overline{r(u, \mathbb{I})}$ denote the obtained estimate of $|\mathcal{R}(u, \mathbb{I})|$. Hence, ATNF approximates $|\mathcal{N}(\mathbb{I})|$ by using the estimates $\overline{r(u, \mathbb{I})}$ as follows:

$$\overline{|\mathcal{N}(\mathbb{I})|} = \sum_{u \in V} \overline{r(u, \mathbb{I})}.$$

By linearity of expectation, the approximation performed by ATNF is unbiased and has relative error bounded by $\epsilon$ with high probability, whenever $k \in \Theta\left(\frac{\log n}{\epsilon^2}\right)$.

## 4. Experimental results

This section is devoted to analyze the performance of ATNF on approximating $\mathcal{N}(\mathbb{I})$. We evaluate both the running time of ATNF and the quality of the approximation achieved, comparing our results with the ones provided by ETNF. Summarizing, the two algorithms we are going to compare are the following.

ATNF Our method for approximating the earliest arrival reverse temporal cones and hence $\mathcal{N}(\mathbb{I})$, as described in Section 3.1. As the quality of the approximation of ATNF and its running time depend on the value of $k$, we analyze the behaviour of ATNF varying $k$ in the set $\{2, 4, 8, 16, 32, 64, 128\}$.

ETNF The method in [9] described in Section 1 to compute exactly the earliest arrival temporal cones and hence $\mathcal{N}(\mathbb{I})$ for any $\mathbb{I}$.

*Implementation and Computing Platform*

Our computing platform is a machine with Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40GHz, 24 virtual cores, 128 Gb RAM, running Ubuntu Linux version 4.4.0-22-generic. The code has been written in Java, using Java 1.8, for both the competitors.

*Dataset*

In order to perform our experiments, we used the following temporal graphs (other temporal networks will be considered in Section 5, in which we will perform a case study based on the directed public transport networks of 25 cities).

- COLLEGE: Private messages sent on an online social network at the University of California, Irvine. An edge $(u, v, t)$ means that user $u$ sent a private message to user $v$ at time $t$ [26,27].
- ENRON: Emails between Enron employees 1999 and 2003. An edge $(u, v, t)$ means that user $u$ sent an email to user $v$ at time $t$.
- FACEBOOK-WALLPOST: A small subset of posts to other user's wall on Facebook. The nodes of the network are Facebook users, and each directed temporal edge represents one post, linking the users writing a post to the user whose wall the post is written on [28–30].
- IMDB: Every node corresponds to an actor and two actors are connected by their collaboration in a movie, where the appearing time of an edge is the year of the movie. We will consider both the whole temporal collaboration graph and the graphs induced by the following genres: adventure, horror, thriller, crime, romance, action, comedy, drama (for each genre, we consider only the edges corresponding to movies classified in that genre) [31].
- ROLLERNET: Opportunistic sighting of Bluetooth devices by groups of rollerbladers carrying Intel iMotes during a roller tour. The 62 iMotes performed neighborhood scans every 15 second [32,33].
- TOPOLOGY: The nodes are autonomous systems and the edges are connections between autonomous systems. The appearing time of an edge is the time-point of the corresponding connection [29,30,34].
- TWITTER: Tweets about the migrant crisis of 2015 [35,36]. A directed edge $(u, v, t)$ means that user $u$ retweeted a tweet of user $v$ at time $t$.
- WIKI-ELECTION: The network of users from the English Wikipedia that voted for and against each other in admin elections. Nodes represent individual users, and edges represent votes. Each edge is annotated with the date of the vote [29,30,37].

The dimensions of the above temporal graphs are summarized in Table 1, where we report for each graph its number of nodes and its number of edges, and whether the graph is directed or not. For a given network $G$, let $t_\alpha$ and $t_\omega$ denote the minimum and maximum appearing time, respectively, of the temporal edges included in $G$. We have then considered different intervals $\mathbb{I} = [t_\alpha, t_\beta]$, for increasing values of $t_\beta$ with $t_\beta \in \mathcal{T}(G)$, where $\mathcal{T}(G)$ is the time horizon of $G$ (that is, the union of all the appearing times of its temporal edges). Both ETNF and ATNF compute (or approximate) the number of pairs of nodes $u$ and $v$ such that, starting from $u$ not before time $t_\alpha$, we can reach $v$ within time $t_\beta$. As a result, we get an exact cumulative frequency distribution running ETNF and its approximation running ATNF. Section 4.1 is devoted to measure the quality of this approximation, while Section 4.2 shows the running times.

## 4.1. Quality of the Approximation

In order to evaluate the quality of the approximation, for each $\mathbb{I} = [t_\alpha, t_\beta]$ we have compared the approximation $\overline{|\mathcal{N}(\mathbb{I})|}$ provided by ATNF with respect to the exact value $|\mathcal{N}(\mathbb{I})|$ provided by ETNF, using the RELATIVE ERROR. We have hence considered the MEAN RELATIVE ERROR (in short, MRE) among all the intervals $\mathbb{I} = [t_\alpha, t_\beta]$ for all values of $t_\beta$ in $\mathcal{T}(G)$. More formally, the MRE is defined as follows:

$$\text{MRE} = \frac{1}{|\mathcal{T}(G)|} \sum_{t_\beta \in \mathcal{T}(G)} \frac{\left| \overline{|\mathcal{N}([t_\alpha, t_\beta])|} - |\mathcal{N}([t_\alpha, t_\beta])| \right|}{|\mathcal{N}([t_\alpha, t_\beta])|}.$$

For each $k \in \{2, 4, 8, 16, 32, 64, 128\}$, we have repeated the experiments ten times and we have reported our results in Table 2. In this table, for each $k$, we have reported the average MRE, denoted as

| GRAPH | MRE of ATNF | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $k = 2$ | | $k = 4$ | | $k = 8$ | | $k = 16$ | | $k = 32$ | | $k = 64$ | | $k = 128$ | |
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| TOPOLOGY | 0.288 | 0.185 | 0.145 | 0.124 | 0.075 | 0.076 | 0.096 | 0.075 | 0.066 | 0.041 | 0.050 | 0.044 | 0.034 | 0.028 |
| ROLLERNET | 0.444 | 0.647 | 0.192 | 0.235 | 0.110 | 0.112 | 0.055 | 0.049 | 0.017 | 0.019 | 0.000 | 0.000 | 0.000 | 0.000 |
| ENRON | 0.578 | 0.481 | 0.454 | 0.505 | 0.161 | 0.118 | 0.138 | 0.136 | 0.075 | 0.070 | 0.069 | 0.063 | 0.046 | 0.041 |
| COLLEGE | 0.531 | 0.406 | 0.450 | 0.537 | 0.199 | 0.138 | 0.118 | 0.098 | 0.108 | 0.076 | 0.043 | 0.033 | 0.028 | 0.025 |
| WIKI-ELECTION | 0.318 | 0.197 | 0.387 | 0.347 | 0.152 | 0.103 | 0.115 | 0.096 | 0.088 | 0.066 | 0.072 | 0.047 | 0.038 | 0.028 |
| FACEBOOK-WALLPOST | 0.430 | 0.348 | 0.370 | 0.251 | 0.213 | 0.156 | 0.168 | 0.110 | 0.064 | 0.059 | 0.067 | 0.045 | 0.058 | 0.041 |
| IMDB-ADVENTURE | 0.350 | 0.236 | 0.280 | 0.206 | 0.182 | 0.130 | 0.135 | 0.115 | 0.087 | 0.067 | 0.044 | 0.039 | 0.047 | 0.041 |
| IMDB-HORROR | 0.419 | 0.371 | 0.220 | 0.155 | 0.191 | 0.194 | 0.134 | 0.104 | 0.079 | 0.081 | 0.056 | 0.066 | 0.029 | 0.027 |
| IMDB-THRILLER | 0.472 | 0.703 | 0.239 | 0.377 | 0.144 | 0.182 | 0.094 | 0.105 | 0.064 | 0.070 | 0.044 | 0.046 | 0.020 | 0.027 |
| IMDB-CRIME | 0.461 | 0.402 | 0.256 | 0.230 | 0.247 | 0.267 | 0.121 | 0.101 | 0.108 | 0.104 | 0.070 | 0.056 | 0.035 | 0.027 |
| IMDB-ROMANCE | 0.546 | 0.496 | 0.395 | 0.387 | 0.131 | 0.103 | 0.122 | 0.093 | 0.081 | 0.079 | 0.089 | 0.077 | 0.043 | 0.034 |
| IMDB-ACTION | 0.420 | 0.345 | 0.275 | 0.197 | 0.184 | 0.143 | 0.136 | 0.097 | 0.085 | 0.063 | 0.064 | 0.054 | 0.043 | 0.034 |
| IMDB-COMEDY | 0.585 | 0.680 | 0.285 | 0.195 | 0.216 | 0.182 | 0.135 | 0.095 | 0.113 | 0.080 | 0.073 | 0.050 | 0.052 | 0.039 |
| IMDB-DRAMA | 0.776 | 1.335 | 0.250 | 0.175 | 0.185 | 0.136 | 0.144 | 0.134 | 0.099 | 0.079 | 0.072 | 0.057 | 0.042 | 0.030 |

**Table 2.** Comparing the quality of the approximation of ATNF with respect to ETNF for approximating $|\mathcal{N}(\mathbb{I})|$. For each $k$, the average $\mu$ and the standard deviation $\sigma$ of the MRE over ten experiments are reported.

$\mu$, and the standard deviation, denoted as $\sigma$, over the ten experiments (note that both $\mu$ and $\sigma$ are 0 for the ROLLERNET network, whenever $k \geq 64$, since the graph has less than 64 nodes and hence ATNF turns out to always compute the exact values).

In general, both $\mu$ and $\sigma$ consistently decrease while increasing $k$. For $k = 2$, 4, and 8, the average MRE appears to be quite large: for $k = 2$, the MRE can be up to 50%, and for $k = 8$ the MRE can be close to 20%. By increasing $k$ to 16 we get an average MRE consistently smaller than 17%, which further reduces with $k = 32$ and $k = 64$, where in both the cases the average MRE is very often below 8%. Finally, for $k = 128$, the average MRE appears to be always smaller than 5.3%. For the sake of completeness, looking at the values of $\sigma$, we can observe how the variability of the experiments is more controlled when $k$ increases.

Note that, in the table, the IMDB-ALL and TWITTER networks are missing since, according to our estimate and as shown in the next section, ETNF would have taken more than one week for the first and more than 190 days for the second to complete. Hence, for these two graphs, no quality comparison was possible.

### *4.2. Running Time and Time Comparison with respect to* ETNF

Table 3 reports the average running time (in milliseconds) of ATNF and of ETNF, respectively, to get the approximate and the exact value of $|\mathcal{N}(\mathbb{I})|$ for each $\mathbb{I} = [t_\alpha, t_\beta]$. Increasing the value of $k$, the running time of ATNF increases consistently (as also the quality of the approximation). In the case of bigger graphs, the running time of ATNF is orders of magnitude smaller with respect to the one of ETNF (rightmost column). We have highlighted this improvement in Table 4, where we have shown the ratio between the running time of ATNF and the one of ETNF.

Concerning the smaller graphs, like ROLLERNET and ENRON, which have respectively 63 and 150 nodes, we have observed that ATNF turns to be exact if $k$ is set to a value greater than this number of nodes. In the case of ROLLERNET, it is interesting to note that the dynamic programming algorithm (which, hence, is exact for $k = 64$ or 128) is strictly faster than the BFS approach implemented by ETNF. On the other hand, as in the case of ENRON, the running time of ATNF, although not exact, can be sometimes worse than the one of ETNF whenever the graph has few nodes.

In the general case, running ATNF with $k = 128$, and thus getting an average MRE below 5.3% (as we have seen in the previous section), the running time of ATNF is very often below 0.3% the running

| GRAPH | Running Time of ATNF | | | | | | | Running Time of |
|---|---|---|---|---|---|---|---|---|
| | $k = 2$ | $k = 4$ | $k = 8$ | $k = 16$ | $k = 32$ | $k = 64$ | $k = 128$ | ETNF |
| TOPOLOGY | 2 419.4 | 2 428.7 | 2 547.8 | 2 937.4 | 4 700.3 | 9 892.4 | 31 295.7 | 1 715 090 |
| ROLLERNET | 498.9 | 740.8 | 1066.5 | 1759.2 | 4079.3 | 1040.3 | 956.2 | 7 636 |
| ENRON | 70.2 | 34.2 | 75.1 | 131.4 | 262.9 | 822.9 | 1 201.5 | 1 172 |
| COLLEGE | 93.5 | 120.9 | 170.7 | 312.5 | 992.9 | 3 010.1 | 9468.7 | 32 394 |
| WIKI-ELECTION | 142.1 | 178.3 | 329.9 | 611.8 | 1 478 | 4 763 | 16 811.1 | 202 519 |
| FACEBOOK-WALLPOST | 1 542.7 | 2 376.9 | 3 256.9 | 6 188.5 | 17 152.6 | 56 847.8 | 172 628.4 | 12 469 653 |
| IMDB-ADVENTURE | 232.9 | 307.2 | 348.4 | 472.2 | 967.8 | 2 428.8 | 6 925.5 | 2 125 739 |
| IMDB-HORROR | 393.6 | 424.7 | 417.4 | 468.1 | 730.5 | 1 352.1 | 3 239.2 | 3 039 487 |
| IMDB-THRILLER | 460.5 | 473.4 | 504.9 | 586.5 | 1 065.6 | 2 517.1 | 6 549.6 | 3 821 616 |
| IMDB-CRIME | 383.1 | 402.6 | 477.0 | 628.6 | 1 315.3 | 3 230.9 | 9 455.8 | 3 745 721 |
| IMDB-ROMANCE | 704.2 | 758.8 | 873.6 | 1 196.2 | 2 160.6 | 5 773.4 | 16 980.6 | 7 046 556 |
| IMDB-ACTION | 865.7 | 927.5 | 1 014.8 | 1 243.6 | 2 645.3 | 6 060.7 | 18 109.9 | 7 079 857 |
| IMDB-COMEDY | 2 825.3 | 3 033.0 | 3 057.8 | 4 075.4 | 7 131.3 | 15 709.5 | 4 2579.4 | 31 806 022 |
| IMDB-DRAMA | 9 619.7 | 9 923.5 | 11 022.8 | 13 050.7 | 19 975.2 | 38 993.5 | 101 945 | 113 236 400 |
| IMDB-ALL | 39 671.1 | 40 273.2 | 41 582.4 | 44 120.5 | 55 290.4 | 91 824.0 | 212 132.1 | †622 185 999 |
| TWITTER | 31 892.7 | 40 758.1 | 61 724.0 | 135 468.9 | 424 497.6 | 1 325 115.7 | 4 222 861.0 | †25 771 840 132 |

**Table 3.** Running times (milliseconds) of ATNF for different values of $k$ compared with the running time of ETNF. The numbers marked with † are estimated due to the computation limits of ETNF.

| GRAPH | Running Time of ATNF over Running Time of ETNF | | | | | | |
|---|---|---|---|---|---|---|---|
| | $k = 2$ | $k = 4$ | $k = 8$ | $k = 16$ | $k = 32$ | $k = 64$ | $k = 128$ |
| TOPOLOGY | 0.00141 | 0.00142 | 0.00149 | 0.00171 | 0.00274 | 0.00577 | 0.01825 |
| ROLLERNET | 0.06534 | 0.09701 | 0.13967 | 0.23038 | 0.53422 | 0.13624 | 0.12522 |
| ENRON | 0.05990 | 0.02918 | 0.06408 | 0.11212 | 0.22432 | 0.70213 | 1.02517 |
| COLLEGE | 0.00289 | 0.00373 | 0.00527 | 0.00965 | 0.03065 | 0.09292 | 0.29230 |
| WIKI-ELECTION | 0.00070 | 0.00088 | 0.00163 | 0.00302 | 0.00730 | 0.02352 | 0.08301 |
| FACEBOOK-WALLPOST | 0.00012 | 0.00019 | 0.00026 | 0.00050 | 0.00138 | 0.00456 | 0.01384 |
| IMDB-ADVENTURE | 0.00011 | 0.00014 | 0.00016 | 0.00022 | 0.00046 | 0.00114 | 0.00326 |
| IMDB-HORROR | 0.00013 | 0.00014 | 0.00014 | 0.00015 | 0.00024 | 0.00044 | 0.00107 |
| IMDB-THRILLER | 0.00012 | 0.00012 | 0.00013 | 0.00015 | 0.00028 | 0.00066 | 0.00171 |
| IMDB-CRIME | 0.00010 | 0.00011 | 0.00013 | 0.00017 | 0.00035 | 0.00086 | 0.00252 |
| IMDB-ROMANCE | 0.00010 | 0.00011 | 0.00012 | 0.00017 | 0.00031 | 0.00082 | 0.00241 |
| IMDB-ACTION | 0.00012 | 0.00013 | 0.00014 | 0.00018 | 0.00037 | 0.00086 | 0.00256 |
| IMDB-COMEDY | 0.00009 | 0.00010 | 0.00010 | 0.00013 | 0.00022 | 0.00049 | 0.00134 |
| IMDB-DRAMA | 0.00008 | 0.00009 | 0.00010 | 0.00012 | 0.00018 | 0.00034 | 0.00090 |
| IMDB-ALL† | 0.00006 | 0.00006 | 0.00007 | 0.00007 | 0.00009 | 0.00015 | 0.00034 |
| TWITTER† | 1.2 E-6 | 1.6 E-6 | 2.4 E-6 | 5.2 E-6 | 0.00002 | 0.00005 | 0.00016 |

**Table 4.** Ratio between the running time of ATNF and the one of ETNF for different values of $k$ (lower is better). The values for the graphs marked with † are estimated due to the computational limits of ETNF.

time of ETNF. This improvement is even more striking for bigger graphs, where the running time of ATNF further reduces to 0.1%.

For the two biggest graphs, i.e. IMDB-ALL and TWITTER, we were not able to run ETNF, due to the large amount of time required by the method. For this reason, we have estimated its running time (reported with † in Table 3). The estimation is based on the fact that ETNF runs $n$ single-source procedures, each one from a different source node. It is possible to estimate the average running time $\mu$ of a single-source procedure, sampling $\Theta(\frac{log n}{\epsilon^2})$ sources and computing the average time $\overline{\mu}$. It is easy to show that this is an unbiased estimator and that, by using the Hoeffding's inequality, $|\mu - \overline{\mu}|$ is bounded with high probability by $\epsilon \cdot r$, where $r$ is an upper bound on the maximum time needed by a single-source procedure, e.g. the time for visiting all the edges. By multiplying $\overline{\mu}$ by $n$, we get an estimation of the running time of ETNF. We have verified experimentally that the order of magnitude reported by our estimation is consistent with the actual time used by ETNF for the smaller graphs. As a

| City | Stops | Edges | Day | R | G | City | Stops | Edges | Day | R | G |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Adelaide | 7 548 | 404 300 | 12/12/16 | 40 | 4 | Belfast | 1 917 | 122 693 | 09/05/16 | 30 | 3 |
| Berlin | 4 601 | 1 048 218 | 04/25/16 | 30 | 3 | Bordeaux | 3 435 | 236 595 | 12/12/16 | 30 | 3 |
| Brisbane | 9 645 | 392 805 | 12/12/16 | 40 | 4 | Canberra | 2 764 | 124 305 | 01/09/17 | 30 | 3 |
| Detroit | 5 683 | 214 863 | 12/12/16 | 30 | 3 | Dublin | 4 571 | 407 240 | 12/12/16 | 20 | 2 |
| Grenoble | 1 547 | 114 492 | 11/14/16 | 20 | 2 | Helsinki | 6 986 | 686 457 | 12/12/16 | 30 | 3 |
| Kuopio | 549 | 32 122 | 12/12/16 | 10 | 1 | Lisbon | 7 073 | 526 179 | 11/21/16 | 30 | 3 |
| Luxembourg | 1 367 | 186 752 | 11/28/16 | 20 | 2 | Melbourne | 19 493 | 1 098 227 | 12/12/16 | 50 | 5 |
| Nantes | 2 353 | 196 421 | 12/12/16 | 20 | 2 | Palermo | 2 176 | 226 215 | 09/22/14 | 20 | 2 |
| Paris | 11 950 | 1 823 872 | 12/12/16 | 35 | 4 | Prague | 5 147 | 670 423 | 12/12/16 | 30 | 3 |
| Rennes | 1 407 | 109 075 | 12/19/16 | 20 | 2 | Rome | 7 869 | 1 051 211 | 11/06/17 | 20 | 2 |
| Sydney | 24 063 | 1 265 135 | 12/19/16 | 50 | 5 | Toulouse | 3 329 | 224 516 | 12/12/16 | 20 | 2 |
| Turku | 1 850 | 133 512 | 12/12/16 | 10 | 1 | Venice | 1 874 | 118 519 | 12/12/16 | 20 | 2 |
| Winnipeg | 5 079 | 333 882 | 12/12/16 | 30 | 3 | | | | | | |

**Table 5.** The 25 cities included in the public transport network dataset [14]. The Stops column indicates the number of nodes, the Edges column the number of temporal edges, the Day column the day in which the data were collected, the R column the city's radius (in km), and the G column the group the city belongs to.

result, applying our estimation for the biggest graphs,[1] ETNF requires 622 185 seconds (more than a week) for IMDB-ALL and 25 771 840 seconds (more than 198 days) for TWITTER. On the other hand, setting $k = 128$, ATNF is able to approximate $|\mathcal{N}(\mathbb{I})|$ in only 112 seconds for the former graphs and in 71 minutes for the latter one, as shown in Table 3. Comparing these running times with our estimates (last two rows of Table 4), ATNF turns out to be four orders of magnitude faster than ETNF.

## 5. Case Study: Comparison of 25 public transport networks

In this section, we will show how the ATNF algorithm for computing the approximation of the temporal neighborhood functions can be applied in order to perform an exhaustive analysis of some reachability properties of several temporal graphs in a very efficient way. In particular, we will make use of a recently published collection of 25 cities' public transport networks [14]. This collection is available in multiple formats including the temporal edge list for a specific day, which is the format we use here. The list of the 25 cities is summarized in Table 5, where, for each city, we provide the number of stops (that is, the number of vertices of the temporal graph), the number of temporal edges, the day in which the data were collected, and the radius $R$ around the city's central point that should cover all the continuous and dense parts of the city and its public transport network [14]. We have grouped the 25 cities into five groups according to the vale of $R$. In particular, group $i$ contains all the cities such that $10(i-1) < R \leq 10i$, for $1 \leq i \leq 5$.

Our goal is to analyze the *reachability efficacy* of each public transport network. To this aim, we compute the (approximate) value of the temporal neighborhood functions corresponding to different time intervals of the day in which the data were collected. In particular, we divide the interval between 6am and 9pm into 30, 15, and 10 intervals of length 30, 60, and 90 minutes, respectively. For each interval $\mathbb{I} = [t_\alpha, t_\omega]$, we compute the approximate value of the temporal neighborhood function $|\mathcal{N}(\mathbb{I})|$ (normalized with respect to the number of all possible pairs of nodes), by using the ATNF algorithm. In the following, we will represent the distributions of these values by *reachability diagrams* such as the one shown in Figure 5, which refers to the city of Adelaide.

Observe that, in the case of transport networks, going from one stop to another, i.e. following an edge, takes some time. Therefore, in our dataset, the temporal edge list contains the starting time

[1] A single-source procedure requires on the average 1.18 seconds for IMDB-ALL and 7.33 seconds for TWITTER, with relative standard deviation respectively of 17.49% and 10.87%.

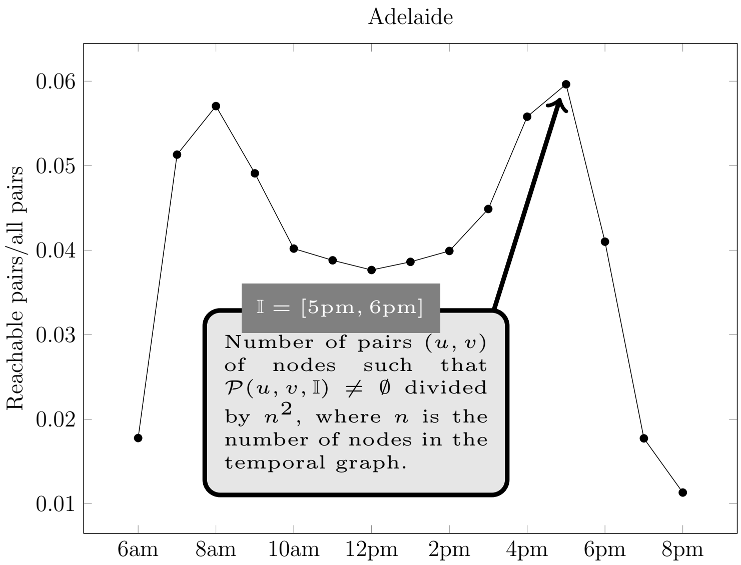

**Figure 5.** The reachability diagram of the public transport network of the city of Adelaide, in which the interval between 6am and 9pm has been split into 15 intervals of one hour each.

$s_e$ and the arrival time $a_e$ of each temporal edge $e$ between two nodes $u$ and $v$. In order to apply our approach, we have used the following transformation. For each edge $e$ in the dataset from $u$ to $v$ with starting time $s_e$ and arrival time $a_e$, with $a_e - s_e > 1$, we have replaced $e$ with a pair of two edges: one from $u$ to $z_e$ with appearing time $s_e$, and the other from $z_e$ to $v$ with appearing time $a_e - 1$, where $z_e$ is a new dummy node. For the edges $e$ from $u$ to $v$ in the dataset such that $a_e - s_e = 1$, we replace $e$ by an edge from $u$ to $v$ with appearing time $s_e$. It is easy to show that there is a one-to-one correspondence between the feasible journeys in the original dataset and the sequences induced by the non-dummy nodes in the temporal paths of the resulting temporal graph. Moreover, it is worth remarking that, when computing the temporal neighborhood function in our transformed temporal graph, we have to focus only on non-dummy nodes. To this aim, we have slightly modified our approach[2] to exclude the dummy nodes from the sketches of our reverse temporal cones and, hence, not to count them in the final estimation of the temporal neighborhood function.

In the left column of Figure 6, we show, for each city group, the reachability diagrams of all cities in the group in the case in which we consider intervals of 60 minutes (these diagrams have been obtained by running 20 times the approximation algorithm with $k = 64$, and by taking the average values). As can be seen, in the case of the first and the last group, the two cities included in the group behave quite similarly, even if the city of Sydney seems to be more efficient in correspondence with the two peaks of its diagram at 8am and at 5pm (which might be considered as the rush hours of the day). In the other three groups, instead, there is clearly a city more efficient than the others of the group (that is, Luxembourg in Group 2, Berlin in Group 3, and Adelaide in Group 4). In each of these groups, the existence of a second more efficient city (that is, Palermo, Winnipeg, and Paris) is also evident. While in Group 2, two cities (that is, Rome and Rennes) fight for the third position, in Group 3 all the other cities are basically equivalent.

Note that we are not evaluating the whole quality of a public transport service, which of course depends on many other factors, such as robustness, safety, reliability, and customer service; instead we are only comparing different public transport networks in terms of their temporal neighborhood functions during a day. In particular, the reachability diagram of a public transport network should only be interpreted as an indicator of the percentage of the pairs of nodes which are connected in

[2] In the base case in Section 3, it is enough to exclude $u$ from $\mathcal{R}(u, [t_\alpha, t_\alpha])$, whenever $u$ is a dummy node.

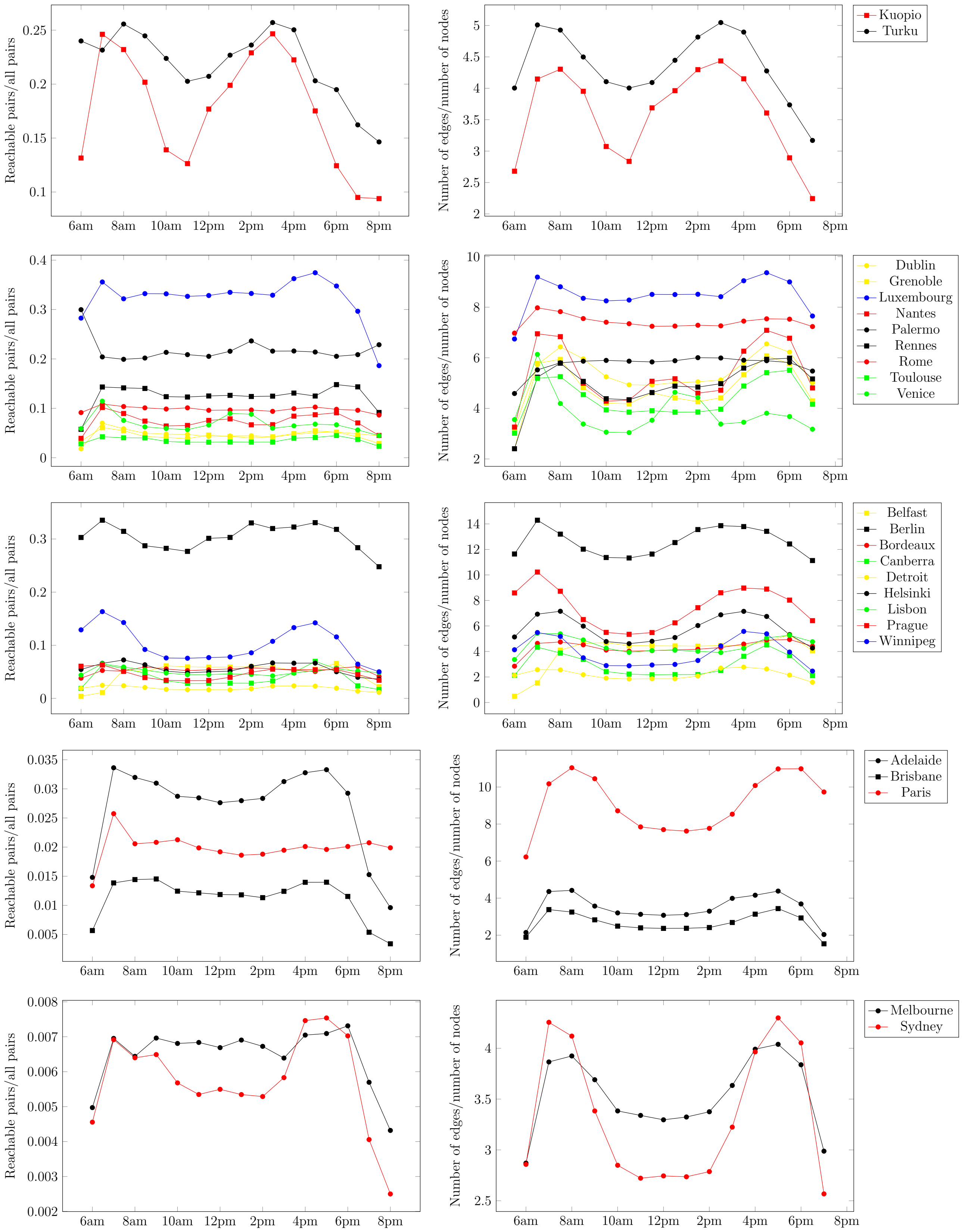

**Figure 6.** The comparison of the reachability diagrams (left) and of the density diagrams (right) of the 25 public transport networks.

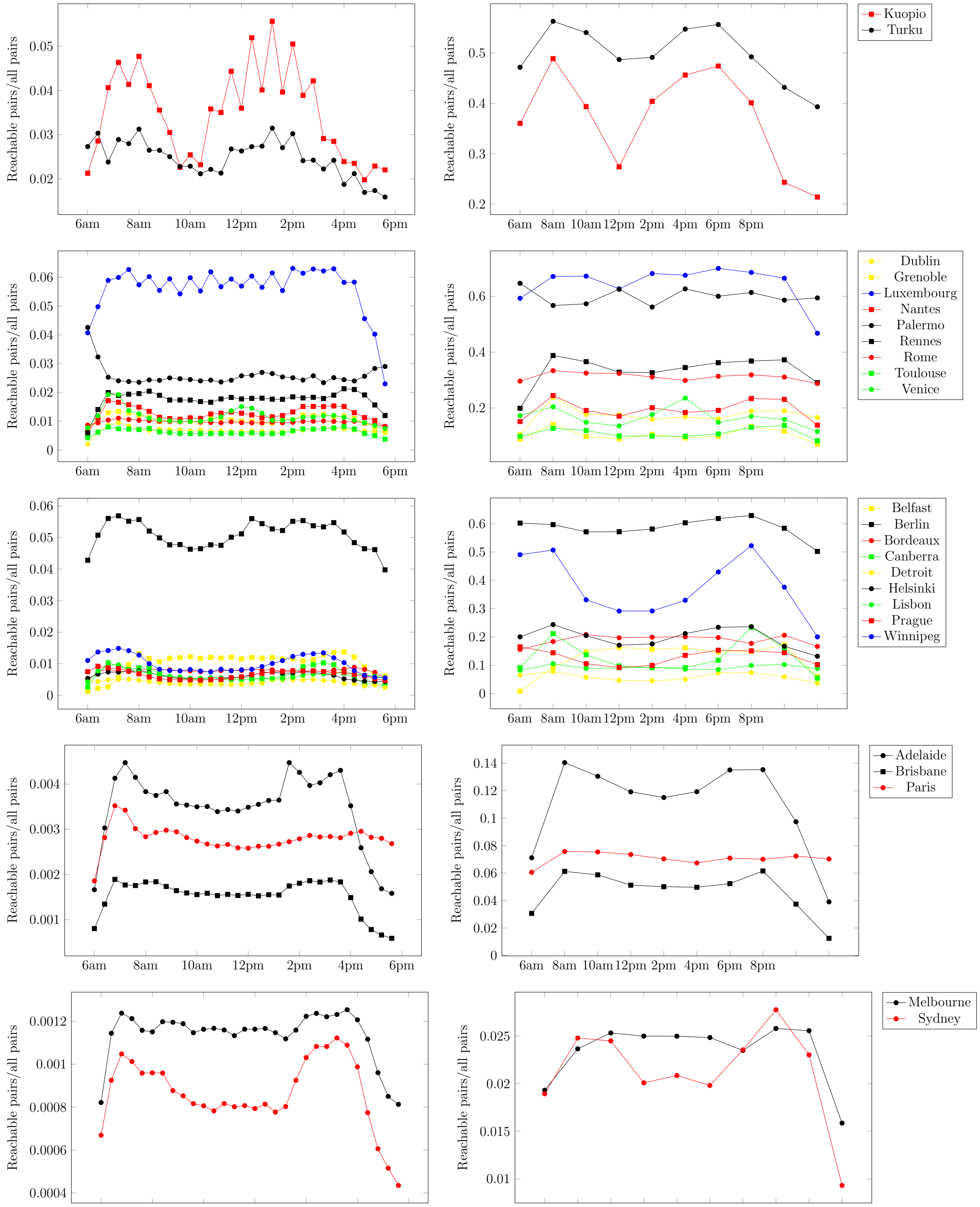

**Figure 7.** The comparison of the reachability diagrams with 30 intervals of 30 minutes each (left), and with 10 intervals of 90 minutes each (right) of the 25 public transport networks.

each interval, with respect to the total number of pairs of nodes. It is also worth emphasizing the fact that, in order to use these diagrams for comparing two public transport networks of two cities, we must implicitly assume that the nodes of a network are uniformly spread in the circle of radius $R$ corresponding to the network itself. It clearly does not make sense to compare a transportation network in which nodes are concentrated in a small sector of the circle with one in which nodes are uniformly spread, since this would turn out into a greater efficiency of the first network with respect to the second one. Unfortunately, we are not able to verify that nodes are uniformly spread in the circle, and we can only assume that this requirement is satisfied.

One could suspect that results similar to the ones shown in the left column of Figure 6 could be obtained by simply considering the temporal density of the original transport networks. For each interval $\mathbb{I} = [t_\alpha, t_\omega]$, let $m(\mathbb{I})$ denote the number of edges from $u$ to $v$ with both starting and arrival time in $\mathbb{I}$. We then define the **temporal density** in the interval $\mathbb{I}$ as $\delta(\mathbb{I}) = \frac{m(\mathbb{I})}{n}$, where $n$ denotes the number of nodes of the network. In the right column of Figure 6, we show, for each city group, the *density diagrams* of all cities included in the group. As it can be seen, in Group 1, 2, 3 and 5, the first city is the same as in the reachability diagrams. In Group 4, instead, Paris seems to have the densest public transport network, even if it is not the first city in terms of reachability cone sizes. Moreover, while the reachability and the density diagrams of Group 1 and 5 seem to be quite consistent, this is not the case for Groups 2 and 3. For instance, in Group 2, Rome is denser than Palermo, but Palermo outperforms Rome in terms of reachability, while in Group 3 this phenomenon happens in the case of Prague and Winnipeg. In other words, the reachability diagram seems to provide some information that is not explicitly given by the density diagram. It is also worth observing how several cities present two peaks in their reachability and density diagrams which we can assume correspond to the two rush hours of the day. This seems to indicate that these cities, in correspondence of these rush hours, increase the number of connections and provide a better service in terms of the number of pairs of connected nodes.

In Figure 7, instead, we compare the reachability diagrams of the five groups obtained with time intervals of length 30 and 90 minutes, respectively (these diagrams should be also compared to the reachability diagrams shown in the left column of Figure 6 which are obtained with time intervals of length 60 minutes). It is worth observing that in the second and in the third group considering shorter time intervals makes the city with the better diagram perform even better compared to the others, and it basically nullifies the differences among the other cities. When larger time intervals are analyzed, it seems that the only effect we get is that the reachability diagrams are systematically shifted up. This is quite reasonable, since we can expect that, in a time interval of one hour and half, a significant percentage of all possible pairs of nodes are now connected. It is anyway worth noting that, in the case of the three cities in Group 4 and of the two cities in Group 5, even with time intervals of 90 minutes,

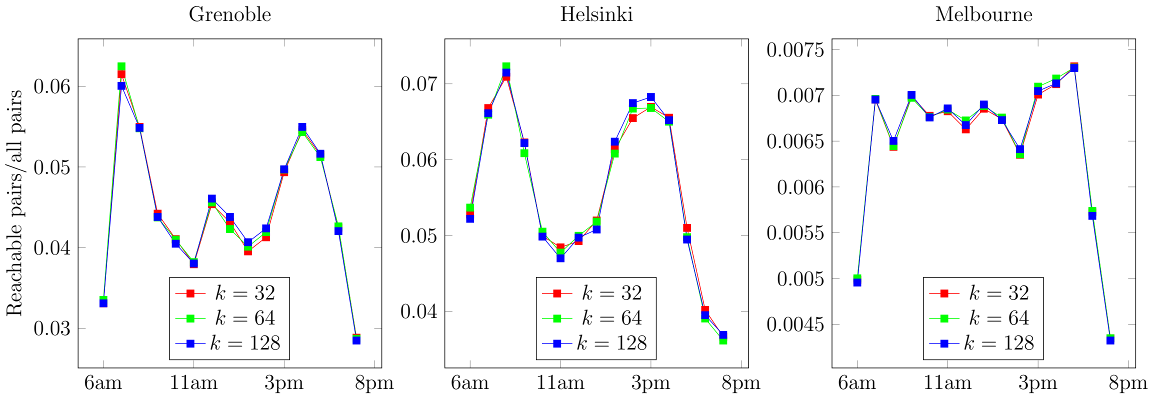

**Figure 8.** The reachability diagrams of three sample cities with different values of $k$.

this percentage is quite low (less than 15% in Group 4 and less than 3% in Group 5). This suggests that in order to obtain a globally almost total connectivity, these cities require trips of quite great duration.

Finally, in Figure 8 we show the reachability diagrams of three sample cities obtained with different values of $k$ in ATNF (the number of experiments is always 20). As it can be seen, the diagrams are almost identical, which might suggest that even $k = 32$ could be used in order to reduce the execution time. However, we decided to use $k = 64$ in the previously described experiments in order to further increase the approximation quality of the results.

As we said at the beginning of this section, the main goal of this analysis has been to describe a case study in which, by using ATNF, we can extremely reduce the total execution time of the experiments. For example, with $k = 64$ and 20 experiments, producing the reachability diagrams of all cities with respect to intervals of length 60 minutes, requires approximately 2.3 hours. The time to get the exact values of the temporal neighborhood functions for each time interval by using ETNF, is instead approximately equal to 68.2 hours, that is, between one and two orders of magnitude bigger than the running time of ATNF, consistently with the results shown in Section 4.

## 6. Conclusions

In this paper, we have proposed a new algorithm for approximating the temporal neighborhood function of a temporal network, which is based on the bottom-$k$ sketch technique. We have experimentally validated the quality and the time performance of our algorithm, by comparing it with a recently proposed scan-based algorithm. We showed that our algorithm is able to obtain good quality results (typically with a mean relative error of 5% or less) in a very small fraction of the time of the exact algorithm (typically 0.3% or less). Finally, we have applied our algorithm to the analysis of the reachability properties of 25 public transport networks and showed that the neighborhood function gives some non-trivial insight about the number of pairs of stops which are reachable from one-another.

From a more technical point of view, a quite natural and straightforward extension of our work consists of applying other sketch techniques, such as, for example, the ones based on the probabilistic counting approach [7,8,38]. We are confident that by using these other techniques our results can be improved both in terms of approximation quality and of execution time. A comparison of the different sketch techniques for approximating the temporal neighborhood function is beyond the scope of this paper, whose main goal is showing how these techniques are extremely effective when applied to temporal networks.

More interestingly, the approach we have described in this paper can also be applied to compute a different kind of temporal cones. Intuitively, these cones would allow us, for each node $u$ of a temporal graph $G = (V, E)$, to determine what is the latest starting time from any other node $v$ that can reach $u$ in a specified interval $\mathbb{I} = [t_\alpha, t_\omega]$, in order to arrive at $u$ not later than $t_\omega$. By appropriately adapting the dynamic programming algorithm for computing the reverse temporal cones, we can show that the latest starting time cones can also be computed in time $O(nm)$ and space $O(n^2m)$ (or $O(nm)$ if we do not need to store all the intermediate results). Moreover, in order to improve the efficiency of the algorithm, we can use, even in this case, the bottom-$k$ sketch technique, exactly as we have done in the case of the reverse temporal cones.

Finally, a more general question is whether other heuristics can be used in order to compute graph metrics for which the sketch approach does not seem to be very efficient in the case of classical graphs. For example, a promising research direction is to define an analogue of the backward breadth-first search for temporal networks, in order to apply techniques which have been extremely powerful for computing the diameter of a graph [39,40] or to apply sampling techniques for computing centrality measures in temporal networks [41,42].

**Author Contributions:** All authors contributed equally to this work.

**Funding:** This research was funded in part by the ANR (French National Agency of Research) under grant ANR-15-CE38-0001 (Algodiv), by Thales within the CLEAR SU-Thales joint lab and by the Ile-de-France program FUI21 under grant 16010629 (iTRAC).

**Acknowledgments:** We thank the anonymous reviewers whose comments helped us to improve this manuscript, in particular for what concerns the way to adapt our algorithm to the case of transportation networks.

**Conflicts of Interest:** The authors declare no conflict of interest. The funding agents had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

1. Holme, P.; Saramäki, J. Temporal networks. *Physics Reports* **2012**, *519*, 97–125.
2. Bhadra, S.; Ferreira, A. Complexity of Connected Components in Evolving Graphs and the Computation of Multicast Trees in Dynamic Networks. ADHOC-NOW, 2003, pp. 259–270.
3. Casteigts, A.; Flocchini, P.; Quattrociocchi, W.; Santoro, N. Time-varying graphs and dynamic networks. *IJPEDS* **2012**, *27*, 387–408.
4. Borgnat, P.; Fleury, E.; Guillaume, J.; Magnien, C.; Robardet, C.; Scherrer, A. Evolving networks. Mining Massive Data Sets for Security, 2007, pp. 198–203.
5. Latapy, M.; Viard, T.; Magnien, C. Stream graphs and link streams for the modeling of interactions over time. *Social Netw. Analys. Mining* **2018**, *8*, 61:1–61:29.
6. Wu, H.; Cheng, J.; Huang, S.; Ke, Y.; Lu, Y.; Xu, Y. Path Problems in Temporal Graphs. *Proceedings of the VLDB Endowment* **2014**, *7*, 721–732. doi:10.14778/2732939.2732945.
7. Palmer, C.R.; Gibbons, P.B.; Faloutsos, C. ANF: a fast and scalable tool for data mining in massive graphs. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada, 2002, pp. 81–90.
8. Boldi, P.; Rosa, M.; Vigna, S. HyperANF: approximating the neighbourhood function of very large graphs on a budget. WWW. ACM, 2011, pp. 625–634.
9. Wu, H.; Huang, Y.; Cheng, J.; Li, J.; Ke, Y. Reachability and time-based path queries in temporal graphs. 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, 2016, pp. 145–156.
10. Cohen, E. Size-Estimation Framework with Applications to Transitive Closure and Reachability. *J. Comput. Syst. Sci.* **1997**, *55*, 441–453.
11. Cohen, E.; Kaplan, H. Tighter estimation using bottom k sketches. *PVLDB* **2008**, *1*, 213–224.
12. Cohen, E. All-Distances Sketches. In *Encyclopedia of Algorithms*; 2016; pp. 59–64.
13. Crescenzi, P.; Marino, A. Degrees of Separation and Diameter in Large Graphs. In *Encyclopedia of Big Data Technologies.*; 2019.
14. Kujala, R.; Weckström, C.; Darst, R.; Madlenocić, M.; Saramäki, J. A collection of public transport network data sets for 25 cities. *Scientific Data* **2018**, *5*. doi:10.1038/sdata.2018.89.
15. Kossinets, G.; Kleinberg, J.M.; Watts, D.J. The structure of information pathways in a social communication network. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008, 2008, pp. 435–443.
16. Huang, S.; Fu, A.W.C.; Liu, R. Minimum Spanning Trees in Temporal Graphs. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data; ACM: New York, NY, USA, 2015; SIGMOD '15, pp. 419–430.
17. Casteigts, A.; Flocchini, P.; Mans, B.; Santoro, N. Shortest, Fastest, and Foremost Broadcast in Dynamic Networks. *Int. Journal of Foundations of Computer Science* **2015**, *26*, 499–522.
18. Bast, H.; Delling, D.; Goldberg, A.V.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; Werneck, R.F. Route Planning in Transportation Networks. In *Algorithm Engineering - Selected Results and Surveys*; 2016; pp. 19–80.
19. Kosowski, A.; Viennot, L. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, 2017, pp. 1462–1478.
20. Li, Q.; Chen, P.W.; Nie, Y.M. Finding optimal hyperpaths in large transit networks with realistic headway distributions. *European Journal of Operational Research* **2015**, *240*, 98 – 108.

21. Gelareh, S.; Nickel, S. Hub location problems in transportation networks. *Transportation Research Part E: Logistics and Transportation Review* **2011**, *47*, 1092 – 1111.
22. Daganzo, C.F. Structure of competitive transit networks. *Transportation Research Part B: Methodological* **2010**, *44*, 434 – 446.
23. Sullivan, J.; Novak, D.; Aultman-Hall, L.; Scott, D. Identifying critical road segments and measuring system-wide robustness in transportation networks with isolating links: A link-based capacity-reduction approach. *Transportation Research Part A: Policy and Practice* **2010**, *44*, 323 – 336.
24. Miller-Hooks, E.; Zhang, X.; Faturechi, R. Measuring and maximizing resilience of freight transportation networks. *Computers & Operations Research* **2012**, *39*, 1633 – 1643.
25. Mishra, S.; Welch, T.F.; Jha, M.K. Performance indicators for public transit connectivity in multi-modal transportation networks. *Transportation Research Part A: Policy and Practice* **2012**, *46*, 1066 – 1085.
26. Panzarasa, P.; Opsahl, T.; Carley, K.M. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* **2009**, *60*, 911–932.
27. Leskovec, J.; Krevl, A. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
28. Viswanath, B.; Mislove, A.; Cha, M.; Gummadi, K.P. On the Evolution of User Interaction in Facebook. Proceedings of the 2Nd ACM Workshop on Online Social Networks, 2009, pp. 37–42.
29. Kunegis, J. KONECT: The Koblenz Network Collection. Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 1343–1350.
30. Institute of Web Science and Technologies. The Koblenz Network Collection. http://konect.uni-koblenz.de.
31. IMDb. IMDb Datasets. http://www.imdb.com/interfaces.
32. Tournoux, P.; Leguay, J.; Benbadis, F.; Whitbeck, J.; Conan, V.; de Amorim, M.D. Density-Aware Routing in Highly Dynamic DTNs: The RollerNet Case. *IEEE Trans. Mob. Comput.* **2011**, *10*, 1755–1768. doi:10.1109/TMC.2010.247.
33. Crawdad. Rollernet dataset. https://crawdad.org/upmc/rollernet/20090202/.
34. Zhang, B.; Liu, R.; Massey, D.; Zhang, L. Collecting the Internet AS-level Topology. *SIGCOMM Comput. Commun. Rev.* **2005**, *35*, 53–61.
35. Borra, E.; Rieder, B. Programmed method: developing a toolset for capturing and analyzing tweets. *Aslib Journal of Information Management* **2014**, *66*, 262 – 278.
36. Borra, E.; Rieder, B. Twitter migrants network. http://data.complexnetworks.fr/Migrants/.
37. Leskovec, J.; Huttenlocher, D.; Kleinberg, J. Governance in Social Media: A Case Study of the Wikipedia Promotion Process. Proc. Int. Conf. on Weblogs and Social Media, 2010.
38. Flajolet, P.; Martin, G.N. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.* **1985**, *31*, 182–209.
39. Crescenzi, P.; Grossi, R.; Imbrenda, C.; Lanzi, L.; Marino, A. Finding the Diameter in Real-World Graphs - Experimentally Turning a Lower Bound into an Upper Bound. ESA, 2010, pp. 302–313.
40. Borassi, M.; Crescenzi, P.; Habib, M.; Kosters, W.A.; Marino, A.; Takes, F.W. Fast diameter and radius BFS-based computation in (weakly connected) real-world graphs: With an application to the six degrees of separation games. *Theor. Comput. Sci.* **2015**, *586*, 59–80.
41. Eppstein, D.; Wang, J. Fast Approximation of Centrality. *J. Graph Algorithms Appl.* **2004**, *8*, 39–45.
42. Magnien, C.; Tarissan, F. Time Evolution of the Importance of Nodes in dynamic Networks. ASONAM, 2015, pp. 1200–1207.