The Next 700 Impossibility Results in Time-Varying Graphs

Nicolas Braud-Santoni[*] Swan Dubois[†] Mohamed-Hamza Kaaouachi[†] Franck Petit[†,‡]

Abstract

We address highly dynamic distributed systems modeled by time-varying graphs (TVGs). We interest in proof of impossibility results that often use informal arguments about convergence. First, we provide a distance among TVGs to define correctly the convergence of TVG sequences. Next, we provide a general framework that formally proves the convergence of the sequence of executions of any deterministic algorithm over TVGs of any convergent sequence of TVGs. Finally, we illustrate the relevance of the above result by proving that no deterministic algorithm exists to compute the underlying graph of any connected-over-time TVG, *i.e.,* any TVG of the weakest class of long-lived TVGs.

1 Introduction

The availability of wireless communications has drastically increased in recent years and established new applications that make various communicating agents and terminals (*e.g.,* robots, sensors, Unmanned Aerial Vehicles, ...) interact together. A common feature of a vast majority of these networks is their *high dynamic*, meaning that their topology keeps continuously changing over time. Classically, distributed systems are modeled by a static undirected connected graph where vertices are processes and edges represent bidirectional communication links. Clearly, such modeling is not suitable for high dynamic networks.

Numerous models taking in account topological changes over time have have been proposed since several decades, to quote only a few, [1, 2, 3, 7, 8, 9, 10]. Some works aim at unifying most of the above approaches. For instance, in [11], the authors introduced the *evolving graphs*. They proposed modeling the time as a sequence of discrete time instants and the system dynamic by a sequence of static graphs, one for each time instant. More recently, another graph formalism, called *Time-Varying Graphs* (TVG), has been provided in [6]. In contrast with evolving graphs, TVGs allow systems evolving within continuous time. Also in [6] and in companion papers [4, 5], TVGs are gathered and ordered into classes depending mainly on two main features: the quality of connectivity among the participating nodes and the possibility/impossibility to perform tasks.

In distributed computing, impossibility results are difficult to prove formally. As an example, *informal* arguments about convergence properties of sequences of objects (*e.g.,* graphs, executions, ...) are often used to prove such results (by building counter examples), that may weaken our confidence in their accuracy. In this paper, we propose a general framework that would help us for proving *formally* impossibility results in TVGs. We first define a metric to compute a distance between any pair of TVGs based on the length of their longest common temporal prefix. Such distance allows to study the convergence of TVG sequences. Our

[*] IAIK, TU Graz, Graz, Austria
[†] Sorbonne Universités, UPMC Université Paris 6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
Inria, Équipe-projet REGAL, F-75005, Paris, France
[‡] Contact author, e-mail: franck.petit@lip6.fr

main result consists in showing that, given an algorithm $\mathcal{A}$ designed for any TVG and a sequence of TVGs that converges toward a TVG g, then the sequence of executions of $\mathcal{A}$ on each TVG of the sequence also converges. Furthermore, the latter converges toward the execution of $\mathcal{A}$ over g.

Next, we provide an example of use of this general result. We consider the weakest class of long-lived TVGs, in the following referred to as *connected-over-time* TVGs, *i.e.,* the class of TVGs where any node can contact any other node infinitely often. It can also be described as the family of TVGs such that the eventual underlying graph (*i.e.,* the subgraph encompassing all edges that are *infinitely often* present) is connected. More precisely, we show that no deterministic algorithm exists to compute the eventual underlying graph of a connected-over-time TVG. This impossibility result intuitively comes from the fact that, with such an algorithm, no node is able to determine whether any of its adjacent edges (appearing/disappearing arbitrarily along the time) may disappear definitively or not.

Section 2 presents the model. Our main result is presented in Section 3, followed by the impossibility result about underlying graph computation over the class of connected-over-time TGVs (Section 4). Section 5 concludes this work.

2 Time-Varying Graph: Model

This section aims to present formally the framework of our study of dynamic systems: time-varying graphs (TVGs). This model was introduced by [6]. We present only definitions needed for the comprehension of our work. We refer the reader to [6] for more details and an interesting taxonomy of TVGs.

Model. Let us first borrow the formalism introduced in [6] in order to describe the distributed systems prone to high dynamic. We consider *distributed systems* made of n computing entities, henceforth indifferently referred to as *nodes*, *vertices*, or *processes*. A process has a local memory, a local sequential and deterministic algorithm, and input/output capabilities. All these entities are gathered in a set V. Let E be a set of edges (or relations) between pairwise entities, that describes interactions between processes, namely communication exchanges. The presence of an edge between two vertices p and q at a given time t means that each vertex among $\{p, q\}$ is able to send a message to the other at t.

The interactions between processes are assumed to take place over a time span $\mathcal{T} \subseteq \mathbb{T}$ called the *lifetime* of the system. The temporal domain $\mathbb{T}$ is generally assumed to be either $\mathbb{N}$ (discrete-time systems) or $\mathbb{R}^+$ (continuous-time systems).

Definition 1 (Time-varying graph [6]). *A time-varying graph (TVG for short) g is a tuple $(V, E, \mathcal{T}, \rho, \zeta, \phi)$ where V is a (static) set of vertices $\{v_1, \ldots, v_n\}$, E a (static) set of edges between these vertices $E \subseteq V \times V$, $\rho : E \times \mathcal{T} \to \{0, 1\}$ (called presence function) that indicates whether a given edge is available (*i.e. *present) at a given time, $\zeta : E \times \mathcal{T} \to \mathbb{T}$ (called edge latency function) indicates the time it takes to cross a given edge if starting at a given date, and $\phi : V \times \mathcal{T} \to \mathbb{T}$ (called process latency function) indicates the time an internal action of a process takes at a given date.*

Given a TVG g, let $\mathcal{T}_g$ be the subset of $\mathcal{T}$ for which a topological event (appearance/disappearance of an edge) occurs in g. The evolution of g during its lifetime $\mathcal{T}$ can be described as the sequence of graphs $\mathcal{S}_g = g_1, g_2, \ldots$, where $g_i = (V, E_i)$ corresponds to the static *snapshot* of g at time $t_i \in \mathcal{T}_g$, *i.e.,* $e \in E_i$ if and only if $\forall t \in [t_i, t_{i+1}[, \rho(e, t) = 1$. Note that, by definition, $g_i \neq g_{i+1}$ for any i.

We consider *asynchronous* distributed systems, *i.e.,* no pair of processes has access to any kind of shared device that could allow to synchronize their execution rate. Furthermore, at any time, no process has access to the output of ζ, *i.e.,* none of them can (*a priori*) predict a bound on the message delay. Note that the ability to send a message to another process at a given time does not mean that this message will be delivered. Indeed, the dynamicity of the communication graph implies that the edge between the two processes may disappear before the delivery of this message leading to the lost of messages in transit.

The presences and absences of an edge are instantly detected by its two adjacent nodes. We assume that our system provides to each process a non-blocking communication primitive named **Send_retry** that ensures the following property. When a process p invokes **Send_retry**(m, q) (where m is an arbitrary message and q another process of V) at time t, this primitive delivers m to q in a finite time provided that there exists a time $t' \geq t$ such that the edge $\{p, q\}$ is present at time t' during at least $\zeta(\{p, q\}, t')$ units of time. In other words, the delivery of the message is ensured if there is, after the invocation of the primitive, an availability of the edge that is sufficient to overcome the communication delay of the edge at this time. Note that this primitive may never deliver a message (*e.g.* if the considered edge never appears after invocation). Details of the implementation of this primitive are not considered here but it typically consists in resending m at each apparition of the edge $\{p, q\}$ until its reception by q. This primitive allows us to abstract from topology changes and asynchronous communication and to write high-level algorithms.

Configurations and executions. The *state* of a process is defined by the values of its variables. Given a TVG g, a *configuration* of g is a vector of $n + 2$ components $(g_i, M_i, p_1, p_2, \dots, p_n)$ such that g_i is a static snapshot of g (*i.e.,* $g_i \in \mathcal{S}_g$), M_i is the set of multi-sets of messages carried over E_i, and p_1 to p_n represent the state of the n processes in V.

An *execution* of the distributed system modeled by g is a sequence of configurations $e = \gamma_0, \dots, \gamma_k, \gamma_{k+1}, \dots$, such that for each $k \geq 0$, during an execution step (γ_k, γ_{k+1}), one of the following event occurs: (i) $g_k \neq g_{k+1}$, or (ii) at least one process receives a message, sends a message, or executes some internal actions changing its state. The *algorithm* executed by g describes the set of all allowed internal actions of processes (in function of their current state or external events as message receptions or time-out expirations) during an execution of g. We assume that during any configuration step (γ_k, γ_{k+1}) of an execution, if $g_k \neq g_{k+1}$, then for each edge e such that $e \in E_k$ and $e \notin E_{k+1}$ (*i.e.,* e disappears during the step (γ_k, γ_{k+1}), none of the messages carried by e belongs to M_{k+1}. Also, for each edge e such that $e \in E_{k+1}$ and $e \notin E_k$ (*i.e.,* e appears during the step (γ_k, γ_{k+1})), e contains no message in configuration γ_{k+1}.

Connected over time TVGs. A key concept of time-varying graphs has been identified in [6]. The authors shows that the classical notion of path in static graphs in meaningless in TVGs. Indeed, some processes may communicate even if there is no (static) path between them at each time. To perform communication between two processes, the existence of a *temporal path* between them is sufficient. They define such a temporal path as follows: a sequence of ordered pairs $\mathcal{J} = \{(e_1, t_1), (e_2, t_2), ..., (e_k, t_k)\}$ such that $\{e_1, e_2, ..., e_k\}$ is a path[1] if for every $i \in [1, k]$, $\rho(e_i, t_i) = 1$ and $t_{i+1} \geq t_i + \zeta(e_i, t_i)$. In other words, a temporal path from process p to process q is a sequence of adjacent edges from p to q such that availability and latency of edges allow the sending of a message from p to q using the **Send_retry** primitive at each intermediate process (refer to [6] for a formal definition). Note that a temporal path is a non symmetric relation between two processes.

Based on various assumptions made about temporal paths (*e.g.,* recurrence, periodicity, symmetry, and so on), the authors propose in [6] proposes a relevant hierarchy of TVG classes. In this paper, we choose to make minimal assumptions on the dynamicity of our system since we restrict ourselves on *connected-over-time* TVGs defined as follows:

[1] A sequence of edges $\{v_1, v'_1\}, \{v_2, v'_2\}, \dots, \{v_k, v'_k\}$ is a *path* if $\forall i \in \{1, k-1\}, v_{i+1} = v'_i$.

Definition 2 (Connected-over-time TVG [6]). *A TVG $(V, E, \mathcal{T}, \rho, \zeta, \phi)$ is connected-over-time if, for any time $t \in \mathcal{T}$ and for any pair of processes p and q of V, there exists a temporal path from p to q after time t. The class of connected-over-time TVGs is denoted by $\mathcal{COT}$*[2].

Note that the lifetime of a connected-over-time TVG is necessarily infinite by definition. The class $\mathcal{COT}$ allows us to capture highly dynamic systems since we only require that any process will be always able to communicate with any other one without any supplementary assumption on this communication (such as delay, periodicity, or used route). In particular, note that a connected-over-time TVG may be disconnected at each time and that the presence of an edge at a given time does not preclude that this edge will appear again after this time. Define an *eventual missing edge* as en edge that appears only a finite number of time during the lifetime of the TVG. The main difficulty encountered in the design of distributed algorithms in $\mathcal{COT}$ is to deal with such eventual missing edges because no process is able to predict if a given adjacent edge is an eventual missing edge or not. Note that the time of the last presence of such an eventual missing edge cannot be even bounded.

Definition 3 ((Eventual) Underlying Graph). *Given a TVG $g = (V, E, \mathcal{T}, \rho, \zeta, \phi)$, the underlying graph of a g is the (static) graph $U_g = (V, E)$. The eventual underlying graph of g is the (static) subgraph $U_g^\omega = (V, E_g^\omega)$ with $E_g^\omega = E \setminus M_g$, where M_g is the set of eventual missing edges of g.*

Intuitively, the underlying graph (sometimes referred to as *footprint*) of a TVG g gathers all edges that appear at least once during the lifetime of g, whereas the eventual underlying graph of g gathers all edges that are infinitely often present during the lifetime of g. Note that, for any TVG of $\mathcal{COT}$, both underlying graph and eventual underlying graph are connected by definition. Let us define the *neighborhood $\mathcal{N}_p$* of a process p is the set of processes with which p shares an edge in the underlying graph.

Induced subclasses. In the following, we focus on specific subclasses of the class $\mathcal{COT}$ to establish our impossibility result. Informally, we focus on subclasses that gather all TVGs whose underlying graph belongs to a given set. The intuition behind this restriction is the following. In practice, some technical reasons may restrict or prevent the communication between some processes, that induces a given underlying graph for the TVG that models our system. In contrast, we cannot predict in general the availabilities and latencies of communication edges, that leads us to consider all TVGs sharing this underlying graph.

Definition 4 (Induced subclass). *Given a set of (static) graphs $\mathcal{F}$ and a class of TVGs $\mathcal{C}$, the subclass of $\mathcal{C}$ induced by $\mathcal{F}$ (denoted by $\mathcal{C}|_\mathcal{F}$) is the set of all TVGs of $\mathcal{C}$ whose underlying graph belongs to $\mathcal{F}$.*

The two following results follow directly from Definitions 2 and 4:

Lemma 1. *In any induced subclass $\mathcal{C}|_\mathcal{F}$, if a TVG $g \in \mathcal{G}$ admits $f \in \mathcal{F}$ as underlying graph, then any other TVG of $\mathcal{C}$ that admits f as underlying graph belongs to $\mathcal{C}|_\mathcal{F}$.*

Lemma 2. *No TVG of $\mathcal{COT}|_\mathcal{F}$ admits an eventual missing edge if and only if $\mathcal{F}$ contains only trees.*

3 Main Theorem

In this section, we state our main result that provides a general framework for proving impossibility results in TVGs. First, we introduce in Section 3.1 some tools needed for the proof of our theorem. Namely, we prove that TVGs and executions sets may be seen as metric spaces with useful topological properties. Then, we prove our main result in Section 3.2.

[2] Authors of [6] refer to this class as C5 in their hiearchy of TVG classes.

3.1 TVG and Output Spaces

TVG Space. For a given time domain $\mathbb{T}$, a given static graph (V, E) and a given latency function ζ, let us consider the set $\mathcal{G}_{(V,E),\mathbb{T},\zeta}$ of all TVGs over $\mathbb{T}$ that admit (V, E) as underlying graph and ζ as latency function. For the sake of clarity, we will omit the subscript $(V, E), \mathbb{T}, \zeta$ and simply denote this set by $\mathcal{G}$. Remark that two distinct TVGs of $\mathcal{G}$ can be distinguished only by their presence function. For any TVG g in $\mathcal{G}$, let us denote its presence function by ρ_g. We define now the following application $d_{\mathcal{G}}$ over $\mathcal{G}$:

$$\begin{array}{rccl} d_{\mathcal{G}}: & \mathcal{G} \times \mathcal{G} & \longrightarrow & [0,1] \\ & (g, g') & \mapsto & \begin{cases} 0 & \text{if } g = g' \\ 2^{-\lambda} & \text{else, with } \lambda = \text{Sup}\,\{t \in \mathbb{T} | \forall t' \leq t, \forall e \in E, \rho_g(e,t') = \rho_{g'}(e,t')\} \end{cases} \end{array}$$

Lemma 3. *The application $d_{\mathcal{G}}$ is an ultrametric over $\mathcal{G}$,* i.e.

1. $\forall (g, g') \in \mathcal{G}^2, d_{\mathcal{G}}(g, g') = 0 \Leftrightarrow g = g'$
2. $\forall (g, g') \in \mathcal{G}^2, d_{\mathcal{G}}(g, g') = d_{\mathcal{G}}(g', g)$
3. $\forall (g, g', g'') \in \mathcal{G}^3, d_{\mathcal{G}}(g, g'') \leq max(d_{\mathcal{G}}(g, g'), d_{\mathcal{G}}(g', g''))$

Proof. The two first properties follow directly from the definition of $d_{\mathcal{G}}$.

To prove the third one, let g, g', and g'' be three TVGs of $\mathcal{G}$. Assume that $d_{\mathcal{G}}(g, g') = 2^{-\lambda'}$ and $d_{\mathcal{G}}(g', g'') = 2^{-\lambda''}$ and let be $\lambda = min(\lambda', \lambda'')$. Then, by definition of $d_{\mathcal{G}}$, we have: $\forall t < \lambda', \forall e \in E, \rho_g(e, t) = \rho_{g'}(e, t)$ and $\forall t < \lambda'', \forall e \in E, \rho_{g'}(e, t) = \rho_{g''}(e, t)$. We can deduce that $\forall t < \lambda, \forall e \in E, \rho_g(e, t) = \rho_{g''}(e, t)$, that means that $d_{\mathcal{G}}(g, g'') \leq 2^{-\lambda}$.

On the other hand, we have: $max(d_{\mathcal{G}}(g, g'), d_{\mathcal{G}}(g', g'')) = max(2^{-\lambda'}, 2^{-\lambda''}) = 2^{-\lambda}$. In conclusion, $d_{\mathcal{G}}(g, g'') \leq max(d_{\mathcal{G}}(g, g'), d_{\mathcal{G}}(g', g''))$, that ends the proof. $\square$

In other words, we can consider $(\mathcal{G}, d_{\mathcal{G}})$ as a metric space (an ultrametric is a particular case of metric) and associate to $(\mathcal{G}, d_{\mathcal{G}})$ the canonical topology, *i.e.* the set of all open balls induced by $d_{\mathcal{G}}$ over $\mathcal{G}$. This topological space have the following property that is useful in the following.

Lemma 4. *The metric space $(\mathcal{G}, d_{\mathcal{G}})$ is complete,* i.e. *a sequence converges in $\mathcal{G}$ if and only if this sequence is Cauchy*[3].

Proof. Recall that, by definition of convergence, any convergent sequence is Cauchy. Hence, let $(g_n)_{n \in \mathbb{N}}$ be a Cauchy sequence in $\mathcal{G}$. We are going to prove that $(g_n)_{n \in \mathbb{N}}$ converges in $\mathcal{G}$. By definition of a Cauchy sequence, we have: $\forall \epsilon \in \mathbb{R}^{*+}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_{\mathcal{G}}(g_k, g_{k+i}) < \epsilon$. In particular, we have: $\forall \lambda \in \mathbb{T}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_{\mathcal{G}}(g_k, g_{k+i}) < 2^{-\lambda}$.

In the other hand, by definition of $d_{\mathcal{G}}$, we know that the existence of $\lambda_{(k,i)} \in \mathbb{T}$ such that $d_{\mathcal{G}}(g_k, g_{k+i}) < 2^{-\lambda_{(k,i)}}$ for $k \in \mathbb{N}$ and $i \in \mathbb{N}$ means that $\forall t < \lambda_{(k,i)}, \forall e \in E, \rho_{g_k}(e, t) = \rho_{g_{k+i}}(e, t)$. Hence, we have: $\forall \lambda \in \mathbb{T}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, \forall t < \lambda, \forall e \in E, \rho_{g_k}(e, t) = \rho_{g_{k+i}}(e, t)$. Let $g_\omega \in \mathcal{G}$ be the TVG defined by $\forall \lambda \in \mathbb{T}, \forall e \in E, \rho_{g_\omega}(e, \lambda) = \rho_{g_k}(e, \lambda)$.

Let $\epsilon \in \mathbb{R}^{*+}$ and λ be the smallest integer such that $2^{-\lambda} < \epsilon$. Then, we know that $\exists k \in \mathbb{N}, \forall i \in \mathbb{N}, \forall t < \lambda, \forall e \in E, \rho_{g_{k+i}}(e, t) = \rho_{g_k}(e, t) = \rho_{g_\omega}(e, t)$. We can deduce that: $\forall i \in \mathbb{N}, d_{\mathcal{G}}(g_k, g_\omega) \leq 2^{-\lambda} < \epsilon$. In other words, $(g_n)_{n \in \mathbb{N}}$ converges to $g_\omega \in \mathcal{G}$, that proves the completeness of $(\mathcal{G}, d_{\mathcal{G}})$. $\square$

[3] Recall that a Cauchy sequence in a metric space (S, d_S) is a sequence $(u_n)_{n \in \mathbb{N}}$ of S whose oscillation converges to 0. More formally, $\forall \epsilon \in \mathbb{R}^{*+}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_S(u_k, u_{k+i}) < \epsilon$

Output Space. For a given algorithm $\mathcal{A}$ and a given TVG g, let us define the $(\mathcal{A}, g)$-output as the function that associates to any time $t \in \mathbb{T}$ the state of g at time t when it executes $\mathcal{A}$. We say that g is the supporting TVG of this output. Let us consider the set $\mathcal{O}_{\mathcal{A},\mathcal{G}}$ of all $(\mathcal{A}, g)$-outputs over all TVGs g of $\mathcal{G}$. For the sake of clarity, we will omit the subscript $\mathcal{A}, \mathcal{G}$ and simply denote this set by $\mathcal{O}$. Remark that two distinct outputs of $\mathcal{O}$ can be distinguished only by their supporting TVG. For any output o in $\mathcal{O}$, let us denote its supporting TVG by g_o. We define now the following application $d_\mathcal{O}$ over $\mathcal{O}$:

$$\begin{array}{rccl} d_\mathcal{O}: & \mathcal{O} \times \mathcal{O} & \longrightarrow & [0,1] \\ & (o, o') & \mapsto & \left\{ \begin{array}{cl} 0 & \text{if } o = o' \\ 2^{-\lambda} & \text{else, with } \lambda = \text{Sup}\ \{t \in \mathbb{T} | \forall t' \leq t, o(t') = o'(t')\} \end{array} \right. \end{array}$$

Due to the similarity between the definition of $d_\mathcal{G}$ and $d_\mathcal{O}$, we can easily prove the following result:

Lemma 5. *The application $d_\mathcal{O}$ is an ultrametric over $\mathcal{O}$.*

As previously, we can consider $(\mathcal{O}, d_\mathcal{O})$ as a metric space, associate to $(\mathcal{O}, d_\mathcal{O})$ the canonical topology and prove the following result:

Lemma 6. *The metric space $(\mathcal{O}, d_\mathcal{O})$ is complete.*

3.2 Convergence of Sequences of TVGs

We are now ready to state our main result. Intuitively, this theorem ensures us that, if we take a sequence of TVGs with ever-growing common prefixes, then the sequence of corresponding outputs also converges. Moreover, we are able to describe the output to which it converges as the output that corresponds to the TVG that shares all commons prefixes of our TVGs sequence. This result is useful since it allows us to construct counter-example in the context of impossibility results. Indeed, it is sufficient to construct a TVG sequence (with ever-growing common prefixes) and to prove that their corresponding outputs violates the specification of the problem for ever-growing time to exhibit an execution that violates infinitely often the specification of the problem. More formally, we have:

Theorem 1. *For any deterministic algorithm $\mathcal{A}$, if a sequence $(g_n)_{n\in\mathbb{N}}$ of $\mathcal{G}$ converges to a given $g_\omega \in \mathcal{G}$, then the sequence $(o_n)_{n\in\mathbb{N}}$ of the $(\mathcal{A}, g_n)$-outputs converges to $o_\omega \in \mathcal{O}$. Moreover, o_ω is the $(\mathcal{A}, g_\omega)$-output.*

Proof. Let $\mathcal{A}$ be a deterministic algorithm and $(g_n)_{n\in\mathbb{N}}$ be a sequence of $\mathcal{G}$ that converges to a given $g_\omega \in \mathcal{G}$. Then, let $(o_n)_{n\in\mathbb{N}}$ be the sequence of the $(\mathcal{A}, g_n)$-outputs.

First, we are going to prove that $(o_n)_{n\in\mathbb{N}}$ converges in $\mathcal{O}$. As $\mathcal{O}$ is complete (see Lemma 6), it is sufficient to prove that $(o_n)_{n\in\mathbb{N}}$ is a Cauchy sequence to obtain this result. Let $\epsilon \in \mathbb{R}^{*+}$. As $\mathcal{G}$ is also complete (see Lemma 4), we know that $(g_n)_{n\in\mathbb{N}}$ is a Cauchy sequence and hence, we have by definition: $\exists k_\epsilon \in \mathbb{N}, \forall i \in \mathbb{N}, d_\mathcal{G}(g_{k_\epsilon}, g_{k_\epsilon+i}) < \epsilon$.

In the other hand, by definition of $d_\mathcal{G}$, we know that the existence of $\lambda_{(k,i)} \in \mathbb{T}$ such that $d_\mathcal{G}(g_k, g_{k+i}) = 2^{-\lambda_{(k,i)}}$ for $k \in \mathbb{N}$ and $i \in \mathbb{N}$ means that $\forall t < \lambda_{(k,i)}, \forall e \in E, \rho_{g_k}(e,t) = \rho_{g_{k+i}}(e,t)$. As $\mathcal{A}$ is deterministic, we can deduce that $\forall t < \lambda_{(k,i)}, o_k(t) = o_{k+i}(t)$ (since $g_{o_n} = g_n$ for any $n \in \mathbb{N}$ by construction of $(o_n)_{n\in\mathbb{N}}$). Then, the definition of $d_\mathcal{O}$ implies that $d_\mathcal{O}(o_k, o_{k+i}) \leq 2^{-\lambda_{(k,i)}}$. In other words, we can deduce that we have $\forall k \in \mathbb{N}, \forall i \in \mathbb{N}, d_\mathcal{O}(o_k, o_{k+i}) \leq d_\mathcal{G}(g_k, g_{k+i})$.

We can conclude that $\exists k_\epsilon \in \mathbb{N}, \forall i \in \mathbb{N}, d_\mathcal{O}(o_{k_\epsilon}, o_{k_\epsilon+i}) < \epsilon$. In conclusion, $(o_n)_{n\in\mathbb{N}}$ is a Cauchy sequence and then converges to $o \in \mathcal{O}$.

Let o_ω be the $(\mathcal{A}, g_\omega)$-output. Then, we are going to prove that $o = o_\omega$. As $d_\mathcal{O}$ is an ultrametric (see Lemma 5), we know that $0 \leq d_\mathcal{O}(o, o_\omega) \leq \max(d_\mathcal{O}(o, o_n), d_\mathcal{O}(o_n, o_\omega))$ for any $n \in \mathbb{N}$. By that precedes, the sequence $(d_{\mathcal{O}}(o, o_n))_{n\in\mathbb{N}}$ converges to 0. Due to the determinism of $\mathcal{A}$ and the completeness of $\mathcal{G}$ and $\mathcal{O}$, we can prove by a similar reasoning as above that $d_{\mathcal{O}}(o_n, o_\omega) \leq d_{\mathcal{G}}(g_n, g_\omega)$ for any $n \in \mathbb{N}$. The convergence of $(g_n)_{n\in\mathbb{N}}$ to g_ω implies that the sequence $(d_{\mathcal{G}}(g_n, g_\omega))_{n\in\mathbb{N}}$ converges to 0. Then, the sequence $(d_{\mathcal{O}}(o_n, o_\omega))_{n\in\mathbb{N}}$ also converges to 0 (since $d_{\mathcal{O}}(o_n, o_\omega) \geq 0$ for any $n \in \mathbb{N}$). Then, the sequence $(\max(d_{\mathcal{O}}(o, o_n), d_{\mathcal{O}}(o_n, o_\omega)))_{n\in\mathbb{N}}$ converges to 0 that implies that $d_{\mathcal{O}}(o, o_\omega) = 0$. As $d_{\mathcal{O}}$ is a metric, we can conclude that $o = o_\omega$, that ends the proof. $\square$

4 Impossibility of Eventual Underlying Graph Computation

In this section, we present an application of our main theorem by proving a natural impossibility result. Namely, we prove that it is impossible to compute the underlying graph of a connected-over-time TVG with a deterministic algorithm. Intuitively, this impossibility result comes from the fact that, with such an algorithm, no process is able to determine if, along its adjacent edges, there exists some eventual missing edges or not. The formal proof of this intuitive result is not as simple as one may think at first glance.

Before presenting the impossibility result, we have to specify our problem. We say that a process p outputs a value v in a configuration γ if one of its variable (called an output variable) has the value v in γ.

Specification 1. *An algorithm $\mathcal{A}$ satisfies the eventual underlying graph specification for a class of TVGs $\mathcal{C}$ if every execution $e = \gamma_0, \gamma_1, \dots$ on any TVG g of $\mathcal{C}$ has a suffix $e_i = \gamma_i, \gamma_{i+1}, \dots$ for a given $i \in \mathbb{N}$ such that each process outputs the eventual underlying graph of g in any configuration of e_i.*

We are now ready to prove the impossibility of eventual underlying graph in connected-over-time TVGs.

Theorem 2. *For any set of (static) graphs $\mathcal{F}$ that does not contain only trees, there exists no deterministic algorithm that satisfies the eventual underlying graph specification for $\mathcal{COT}|_{\mathcal{F}}$.*

Proof. We define, for any TVG $g = (V, E, \mathcal{T}, \rho, \zeta, \phi)$, the TVG $g \oplus \{(e_1, \mathcal{T}_{e_1}), \dots, (e_k, \mathcal{T}_{e_k})\}$ (with, for any $i \in \{0, \dots, k\}$, $e_i \in E$ and $\mathcal{T}_{e_i} \subseteq \mathcal{T}$) as the TVG $(V, E, \mathcal{T}, \rho', \zeta, \phi)$ with:

$$\rho'(e,t) = \begin{cases} 1 \text{ if } \exists i \in \{0, \dots, k\}, e = e_i \text{ and } t \in \mathcal{T}_{e_i} \\ \rho(e,t) \text{ otherwise} \end{cases}$$

By contradiction, assume that there exists a set of (static) graphs $\mathcal{F}$ that does not contain only trees such that there exists a deterministic algorithm $\mathcal{A}$ that satisfies the eventual underlying graph specification for $\mathcal{COT}|_{\mathcal{F}}$. In consequence, any process that executes $\mathcal{A}$ outputs a (static) graph at any time.

By Lemma 2, we know that there exists $g \in \mathcal{COT}|_{\mathcal{F}}$ such that $g = (V, E, \mathcal{T}, \rho, \zeta, \phi)$ admits at least one eventual missing edge e. We construct then a sequence $(g_n)_{n\in\mathbb{N}}$ of TVGs as follows. We set $g_0 = g$ and we define inductively g_i for any $i \in \mathbb{N}$ as follows—refer to Figure 1:

1. Consider the execution of $\mathcal{A}$ over g_i and let $\eta_i \in \mathcal{T} \cup \{+\infty\}$ be the largest time where e belongs to the graph outputted by some process of V (remark that $\eta_i = +\infty$ if and only if e belongs infinitely often to the outputted graph of at least one process);

2. Let $g'_i = g_i \oplus (e, \mathcal{T} \cap]\eta_i, +\infty[)$;

3. Consider the execution of $\mathcal{A}$ over g'_i and let $\alpha_i \in \mathcal{T} \cup \{+\infty\}$ be the smallest time strictly greater than η_i where e belongs to the graph outputted by all process of V (remark that $\alpha_i = +\infty$ if and only if e never belongs simultaneously to the outputted graph of all processes $\eta_i = +\infty$);

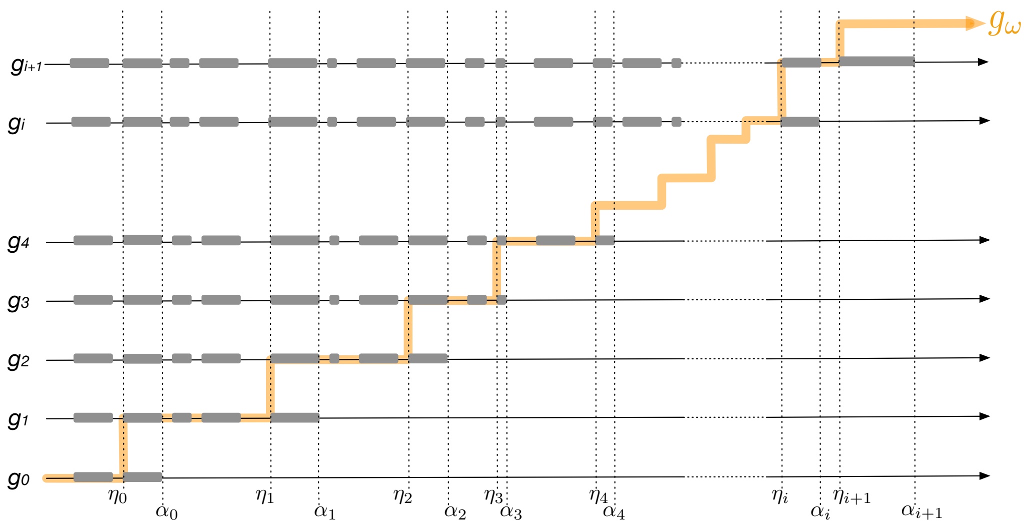

Figure 1: Construction of $(g_n)_{n\in\mathbb{N}}$ in the proof of Theorem 2. Grey bold lines represent instants where e belongs to the graph outputted by all process of V.

4. Let $g_{i+1} = g_i \oplus (e, \mathcal{T} \cap]\eta_i, \alpha_i[)$.

We can prove that, for any $i \in \mathbb{N}$, if g_i belongs to $\mathcal{COT}|_\mathcal{F}$ and if e is an eventual missing edge in g_i, then $\eta_i \neq +\infty$ and $\alpha_i \neq +\infty$. Indeed, assume that e is an eventual missing edge in $g_i \in \mathcal{COT}|_\mathcal{F}$ for a given $i \in \mathbb{N}$. By definition, e does not belong to $U^\omega_{g_i}$. As $\mathcal{A}$ satisfies the eventual underlying graph specification for $\mathcal{COT}|_\mathcal{F}$, we know that e cannot belongs infinitely often to the outputted graph of a process in the execution of $\mathcal{A}$ over g_i, *i.e.,* $\eta_i \neq +\infty$. Then, as e is not an eventual missing edge in g'_i by construction, e belongs to $U^\omega_{g'_i}$. By Lemma 1, g'_i belongs to $\mathcal{COT}|_\mathcal{F}$ since g_i and g'_i share the same underlying graph U_g. As $\mathcal{A}$ satisfies the eventual underlying graph specification for $\mathcal{COT}|_\mathcal{F}$, we know that e belongs eventually to the outputted graph of all processes of V, *i.e.,* $\alpha_i \neq +\infty$.

We obtain that, for any $i \in \mathbb{N}$, if g_i belongs to $\mathcal{COT}|_\mathcal{F}$ and if e is an eventual missing edge in g_i, then g_{i+1} belongs to $\mathcal{COT}|_\mathcal{F}$ and e is an eventual missing edge in g_{i+1}. Indeed, g_{i+1} belongs to $\mathcal{COT}|_\mathcal{F}$ by Lemma 1 (since g_i and g_{i+1} share the same underlying graph U_g). As we proved that $\eta_i \neq +\infty$ and $\alpha_i \neq +\infty$ when e is an eventual missing edge in g_i, g_{i+1} is obtained by adding e during a finite amount of time to g_i, that implies that e is an eventual missing edge in g_{i+1}.

Now, it is sufficient to note that g belongs to $\mathcal{COT}|_\mathcal{F}$ by assumption and that e is an eventual missing edge in $g_0 = g$ by construction to obtain that $(g_n)_{n\in\mathbb{N}}$ is a sequence of $\mathcal{COT}|_\mathcal{F}$ such that $\eta_i \neq +\infty$ and $\alpha_i \neq +\infty$ for any $i \in \mathbb{N}$. Moreover, note that, for any $i \in \mathbb{N}$, $\eta_i < \alpha_i$ (by construction) and $\alpha_i < \eta_{i+1}$ (since e belongs to the graph outputted by any process at time α_i in g_{i+1} whereas e does not belong to the graph outputted by any process at time η_{i+1} in g_{i+1}).

That allows us to define the following TVG: $g_\omega = g \oplus \{(e, \mathcal{T} \cap]\eta_i, \alpha_i[) | i \in \mathbb{N}\}$. Note that $U_{g_\omega} = U_g$ and then, by Lemma 1, that g_ω belongs to $\mathcal{COT}|_\mathcal{F}$. Observe that, for any $k \in \mathbb{N}^*$, we have $d_\mathcal{G}(g_k, g_\omega) = 2^{-\eta_k}$ by construction of $(g_n)_{n\in\mathbb{N}}$ and g_ω. Thus, $(g_n)_{n\in\mathbb{N}}$ converges in $\mathcal{COT}|_\mathcal{F}$ to g_ω.

We are now in measure to apply our main theorem (see Theorem 1) that states that the $(\mathcal{A}, g_\omega)$-output is the limit of the sequence of the $(\mathcal{A}, g_n)$-outputs. In other words, the $(\mathcal{A}, g_\omega)$-output shares a prefix of length η_i with the $(\mathcal{A}, g_i)$-output for any $i \in \mathbb{N}$ (recall that the sequence of the $(\mathcal{A}, g_n)$-outputs is Cauchy since it converges). That means that there exists infinitely many configurations in the execution of $\mathcal{A}$ on g_ω where e

belongs to the outputted graph of all process and infinitely many configurations in the execution of $\mathcal{A}$ on g_ω where e does not belong to the outputted graph of any process, that contradicts the fact that $\mathcal{A}$ satisfies the eventual underlying graph specification for $\mathcal{COT}|_\mathcal{F}$ and ends the proof. $\square$

5 Conclusion

We gave a general framework for providing impossibility results in time-varying graphs. This framework is useful to legitimate informal arguments about convergence of sequences of objects in this context. We used the above result to prove that no deterministic algorithm exists to compute the underlying graph of any connected-over-time TVG. Our general framework is devoted to be used with a large number of problems in TVGs, *e.g.,* overlay construction.

References

[1] A. Anagnostopoulos, R. Kumar, M. Mahdian, E. Upfal, and F. Vandin. Algorithms on evolving graphs. In *ITCS*, pages 149–160, 2012.

[2] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *PODC*, pages 278–281, 1984.

[3] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *SIROCCO*, pages 126–140, 2009.

[4] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic computations in time-varying graphs: Broadcasting under unstructured mobility. *Theoretical Computer Science*, pages 111–124, 2010.

[5] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. Technical report, arXiv:1210.3277, 2012.

[6] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[7] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM – CATAPCC*, pages 27–34, 2003.

[8] A. Ferreira. Building a reference combinatorial model for manets. *Network*, 18(5):24–29, 2004.

[9] A. Ferreira, A. Goldman, and J. Monteiro. On the evaluation of shortest journeys in dynamic networks. In *NCA*, pages 3–10, 2007.

[10] J. Schneider and R. Wattenhofer. Coloring unstructured wireless multi-hop networks. In *PODC*, pages 210–219, 2009.

[11] B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.