The Next 700 Impossibility Results in Time-Varying Graphs

Nicolas Braud-Santoni, Swan Dubois, Mohamed Hamza Kaaouachi, Franck Petit

The Next 700 Impossibility Results in Time-Varying Graphs*†

Nicolas Braud-Santoni[1], Swan Dubois[2], Mohamed-Hamza Kaaouachi[2], and Franck Petit[2]

Abstract

We consider highly dynamic distributed systems modelled by time-varying graphs (TVGs). We first address proof of impossibility results that often use informal arguments about convergence. 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. Next, we focus of the weakest class of long-lived TVGs, *i.e.*, the class of TVGs where any node can communicate any other node infinitely often. We illustrate the relevance of our result by showing that no deterministic algorithm is able to compute various distributed covering structure on any TVG of this class. Namely, our impossibility results focus on the eventual footprint, the minimal dominating set and the maximal matching problems.

1 Introduction

The availability of wireless communications has drastically increased in recent years and established new applications. Humans, agents, devices, robots, and applications interact together through more and more heterogeneous infrastructures, such as mobile *ad hoc* networks (MANET), vehicular networks (VANET), (mobile) sensor and actuator networks (SAN), body area networks (BAN), as well as always evolving network infrastructures on the Internet. In modern networks, items (users, links, equipments, *etc.*) may join, leave, or move inside the network at unforeseeable times. A common feature of these networks is their *high dynamics*, meaning that their topology keeps continuously changing over time. Dynamics, heterogeneity of devices, usages, and participants, and often the unprecedented scale to consider, make the design of such infrastructures extremely challenging. For a vast majority of them, the dynamics are also unpredictable. Classically, distributed systems are modeled by a static undirected connected graph where vertices are processes (nodes, servers, processors, etc.) and edges represent bidirectional communication links. Clearly, such modeling is not suitable for highly dynamic networks.

Numerous models taking topological changes over time into account have been proposed since several decades, *e.g.* [1, 2, 8, 9]. Some works aim at unifying most of the above approaches. For instance, in [12], the authors introduced the *evolving graphs*. They proposed modeling the time as a sequence of discrete time instants and the system dynamics 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 [5]. In contrast with evolving graphs, TVGs allow systems evolving within continuous time. Also in [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.

*This paper is an extended version of [3] and includes materials from [7]. The title is a tribute to a series of papers including "The Next 700 BFT Protocols" (ACM TOCS, 2015) and "The Next 700 Programming Languages" (CACM, 1966) due to the common point between them: the genericity of proposed results.

†This work was performed within the Labex SMART, supported by French state funds managed by the ANR within the "Investissements d'Avenir" programme under reference ANR-11-LABX-65.

[1]IAIK, Graz University of Technology, Austria

[2]Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, LIP6 UMR 7606, France

In this paper, we propose a generic framework that would help to *formally* prove 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 main result consists of 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.

This result is useful for proving impossibility results in the following way. We proceed by contradiction and we assume that there exists an algorithm $\mathcal{A}$ solving the considered problem. Assume now that we are able to build a sequence of TVGs sharing ever-growing common temporal prefixes such that the execution of $\mathcal{A}$ does not converge on these prefixes. Then, by applying our main result, we can deduce that this sequence converges to a TVG G such that the execution of $\mathcal{A}$ on G never converges. In other words, our main result allows to formally construct a counter-example based on a limit of a sequence of TVG. Obviously, the delicate part of the impossibility proof is then delegated to the building of the adequate sequence of TVG depending on the desired impossibility result.

Next, we illustrate the use of this general result by providing various examples of use. We consider the weakest class of long-lived TVGs, in the following referred to as *connected-over-time* ($\mathcal{COT}$, for short) 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.

Within this model, we first show that no deterministic algorithm exists to compute the eventual underlying graph. Intuitively, this impossibility result 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.

Next, we address two well-known and well-studied cover problems in distributed systems, namely the *Maximal Matching* (MM) and the *Minimal Dominating Set* (MDS). In static graphs, MM is defined as follows: A matching M is a set of pairwise non-adjacent edges, that is no two edges share a common vertex. M is maximal if any edge that is not in M is added to M, then M is no longer a matching. Maximal matching has many applications in distributed applications. Examples include among others the problem of assigning tasks to workers or creating pairs of entities. Still in static environment, a dominating set is a subset of vertices of a graph such as each vertex of this graph is either in the dominating set or neighbor of a vertex in the dominating set. A *minimal* dominating set is such that none of its proper subsets is also a dominating set of the graph. Minimal Dominating Set is a key building block for numerous network protocols, *e.g.* hierarchical routing and clustering, multicast, topology control, media access coordination, to name only a few.

Both problems received some attention in the context of dynamic networks, *e.g.* [6, 10, 11]. The difficulty to define covering structures in dynamic networks (including MM and MDS) is pointed out in [4]. Indeed, the authors show that the definition of such structures may become ambiguous, incorrect, or even irrelevant when applied in dynamic systems. As an example, consider the MDS problem. If the dynamics of the graph is modeled as a sequence of static graphs and a new MDS is computed at each topological change as in [11], the stability of the MDS fully depends on the dynamic rate of the network (*i.e.* the relative speed of appearance/disappearance of edges). This natural definition may hence lead to a high instability (or even impossibility of use) of the MDS.

As a consequence, we first provide new definitions for Maximal Matching and Minimal Dominating Set constructions that are relevant in $\mathcal{COT}$ TVGs. More precisely, we require the cover set (either minimal dominating set or maximal matching) to be built over the eventual underlying graph. This requirement ensures us that the cover set will be eventually stable and that each vertex will infinitely often satisfy the local property of the cover set.

In this context, we then use our main result to prove a necessary topological condition for the existence of a deterministic algorithm for both problems in connected-over-time TVGs. The sufficiency of these conditions fall outside the scope of this paper and is proved in [7].

The paper is organized as follows. Section 2 presents the model. Our main result is presented in Section 3,

followed by its three applications about underlying graph computation, minimal dominating set, and maximal matching over the class of connected-over-time TVGs (Section 4). Section 5 concludes this work.

2 Model

This section aims to formally present the framework of our study of dynamic systems: time-varying graphs (TVGs). This model was introduced by [5]. We present only definitions needed for the comprehension of our work. We refer the reader to [5] for more details and an interesting taxonomy of TVGs.

2.1 Model

Let us first borrow the formalism introduced in [5] in order to describe the distributed systems prone to high dynamics. We consider *distributed systems* made of n *processes*. A process has a local memory, a local sequential and deterministic algorithm, and message exchange capabilities. We assume that each process has a unique identifier and that process identifiers are totally ordered by some relation $<$. All these processes are gathered in a set V. Let E be a set of edges (or relations) between pairwise processes, that describes interactions between processes, namely communication exchange. The presence of an edge between two processes p and q at a given time t means that each process among $\{p, q\}$ is able to send a message to the other at t. For any given (static) graph G, we denote by $diam(G)$ the diameter of G (that is, the longest distance between two processes of G).

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 [5]) *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 processes $\{v_1, \ldots, v_n\}$, E a (static) set of edges between these processes $E \subseteq V \times V$, $\rho : E \times \mathcal{T} \to \{0, 1\}$ (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.*

In order to simplify some definitions, we restrict ourselves this definition in the following way.

Definition 2 (Well-formed TVG) *A well-formed TVG $G = (V, E, \mathcal{T}, \rho, \zeta, \phi)$ is a TVG satisfying the two following properties:*

$$\begin{cases} \forall e \in E, \exists t \in \mathcal{T}, \rho(e, t) = 1 \\ \forall e \in E, \forall t \in \mathcal{T}, \exists \epsilon \in \mathcal{T}, (\rho(e, t) = 1 \wedge \forall t' \in]t - \epsilon, t[, \rho(e, t') = 0) \Rightarrow \forall t' \in [t, t + \zeta(e, t)], \rho(e, t') = 1 \end{cases}$$

Intuitively, these properties ensure that any edge of E appears at least once during the lifetime of the TVG and that each apparition of an edge is sufficiently long to overcome its latency at the time of its apparition. In the sequel of this paper, we focus only on well-formed TVGs and we hence omit the words "well-formed" for lightness.

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 $G_i \neq G_{i+1}$ 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 dynamics 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 processes. 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 $(G_i, M_i, p_1, p_2, \ldots, p_n)$ such that G_i is a static snapshot of G (*i.e.* $G_i \in \mathcal{S}_G$), M_i is the set of messages carried by each edge of E_i (one multi-set of messages per edge), and p_1 to p_n represent the state of the n processes in V. We say that a process p *outputs* a value v in a configuration γ if one of its variable (then called an *output variable*) has the value v in γ.

An *execution* of the distributed system modeled by G is a sequence of configurations $e = \gamma_0, \ldots, \gamma_k, \gamma_{k+1}, \ldots$, 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 [5]. 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* (*a.k.a. journey*) between them is sufficient. They define such a temporal path of a TVG G as a sequence of ordered pairs $\{(p_1, t_1), (p_2, t_2), ..., (p_{k-1}, t_{k-1}), (p_k, t_k)\}$ such that $p_1, p_2, \ldots, p_{k-1}, p_k$ is a (static) path of (V, E) and, for every $i \in \{1, \ldots, k-1\}$, $\rho(\{p_i, p_{i+1}\}, t_i) = 1$ and $t_{i+1} \geq t_i + \zeta(\{p_i, p_{i+1}\}, t_i) + \phi(p_{t+1}, t_i + \zeta(\{p_i, p_{i+1}\}, t_i))$. 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 and processes allow the sending of a message from p to q using the **Send_retry** primitive at each intermediate process (refer to [5] for a formal definition). Note that the existence of a temporal path is a non symmetric relation between two processes, even though the graph may be undirected. Based on various assumptions made about journeys (*e.g.* recurrence, periodicity, symmetry, and so on), [5] proposes a relevant hierarchy of TVG classes. In this paper, we choose to make minimal assumptions on the dynamics of our system since we restrict ourselves on *connected-over-time* TVGs defined as follows:

Definition 3 (Connected-over-time TVG [5]) *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 journey from p to q after time t. The class of connected-over-time TVGs is denoted by $\mathcal{COT}$*[1].

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

[1] Authors of [5] refer to this class as C5 in their hierarchy of TVG classes.

to communicate with any other one without any extra 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 an edge that is never present after some finite time. 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 even be bounded.

Definition 4 ((Eventual) Underlying Graph) *The underlying graph of a TVG $G = (V, E, \mathcal{T}, \rho, \zeta, \phi)$ 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 of communication edges, that leads us to consider all TVGs sharing this underlying graph.

Definition 5 (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 3 and 5:

Lemma 1 *In any induced subclass $\mathcal{C}|_\mathcal{F}$, if a TVG 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.

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 $\Gamma_{(V,E),\mathbb{T},\zeta}$ 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 Γ. Remark that two distinct TVGs of Γ can be distinguished only by their presence function. For any TVG G in Γ, let us denote its presence function by ρ_G. We define the application $d_\Gamma : \Gamma \times \Gamma \longrightarrow [0, 1]$ by:

$$(G, G') \quad \mapsto \quad \begin{cases} 0 & \text{if } G = G' \\ 2^{-\lambda} & \text{else, with } \lambda = \text{Sup } \{t \in \mathbb{T} | \forall t' \leq t, \forall e \in E, \rho_G(e, t') = \rho_{G'}(e, t')\} \end{cases}$$

Lemma 3 *The application d_Γ is an ultrametric over Γ,* i.e.

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

Proof. The two first properties follow directly from the definition of d_Γ.

To prove the third one, let G, G', and G'' be three TVGs of Γ. Assume that $d_\Gamma(G, G') = 2^{-\lambda'}$ and $d_\Gamma(G', G'') = 2^{-\lambda''}$ and let be $\lambda = min(\lambda', \lambda'')$. Then, by definition of d_Γ, 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 $d_\Gamma(G, G'') \leq 2^{-\lambda}$.

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

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

Lemma 4 *The metric space (Γ, d_Γ) is complete,* i.e. *a sequence converges in Γ if and only if this sequence is Cauchy*[2].

Proof. Let $(G_n)_{n\in\mathbb{N}}$ be a Cauchy sequence in Γ. By definition of convergence, any convergent sequence is Cauchy; hence, if suffices to prove that $(G_n)_{n\in\mathbb{N}}$ converges in Γ.

By definition of a Cauchy sequence, we have: $\forall \epsilon \in \mathbb{R}^{*+}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_\Gamma(G_k, G_{k+i}) < \epsilon$. In particular, we have: $\forall \lambda \in \mathbb{T}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_\Gamma(G_k, G_{k+i}) < 2^{-\lambda}$.

In the other hand, by definition of d_Γ, we know that the existence of $\lambda_{(k,i)} \in \mathbb{T}$ such that $d_\Gamma(G_k, G_{k+i}) < 2^{-\lambda_{(k,i)}}$ 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 $G_\omega \in \Gamma$ 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: $\forall i \in \mathbb{N}, d_\Gamma(G_k, G_\omega) \leq 2^{-\lambda} < \epsilon$. In other words, $(G_n)_{n\in\mathbb{N}}$ converges to $G_\omega \in \Gamma$, that proves the completeness of (Γ, d_Γ). □

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 $\mathcal{O}_{\mathcal{A},\Gamma}$ of all $(\mathcal{A}, G)$-outputs over all TVGs G of Γ. For the sake of clarity, we will omit the subscript $\mathcal{A}, \Gamma$ 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 the application $d_\mathcal{O} : \mathcal{O} \times \mathcal{O} \longrightarrow [0, 1]$ by:

$$(o, o') \quad \mapsto \quad \begin{cases} 0 & \text{if } o = o' \\ 2^{-\lambda} & \text{else, with } \lambda = \text{Sup } \{t \in \mathbb{T} | \forall t' \leq t, o(t') = o'(t')\} \end{cases}$$

Due to the similarity between the definition of d_Γ 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 now consider $(\mathcal{O}, d_\mathcal{O})$ as a metric space, associate the canonical topology to $(\mathcal{O}, d_\mathcal{O})$, and prove the following result:

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

[2]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, $\forall \epsilon \in \mathbb{R}^{*+}, \exists k \in \mathbb{N}, \forall i \in \mathbb{N}, d_S(u_k, u_{k+i}) < \epsilon$.

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 Γ converges to a given $G_\omega \in \Gamma$, 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 Γ that converges to a given $G_\omega \in \Gamma$. 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 Γ 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: $\exists k_\epsilon \in \mathbb{N}, \forall i \in \mathbb{N}, d_\Gamma(G_{k_\epsilon}, G_{k_\epsilon+i}) < \epsilon$.

In the other hand, by definition of d_Γ, we know that the existence of $\lambda_{(k,i)} \in \mathbb{T}$ such that $d_\Gamma(G_k, G_{k+i}) = 2^{-\lambda_{(k,i)}}$ 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 $\forall k \in \mathbb{N}, \forall i \in \mathbb{N}, d_\mathcal{O}(o_k, o_{k+i}) \leq d_\Gamma(G_k, G_{k+i})$.

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 Γ and $\mathcal{O}$, we can prove by a similar reasoning as above that $d_\mathcal{O}(o_n, o_\omega) \leq d_\Gamma(G_n, G_\omega)$ for any $n \in \mathbb{N}$. The convergence of $(G_n)_{n\in\mathbb{N}}$ to G_ω implies that the sequence $(d_\Gamma(G_n, G_\omega))_{n\in\mathbb{N}}$ 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. □

4 Applications

In this section, we present some applications of our main theorem by proving impossibility results on three classical problems on the very weak class of connected-over-time TVGs. As the proofs of these impossibility results strongly rely on our main theorem, they naturally make use of similar arguments. Nevertheless, the construction of the ad-hoc sequence of TVGs is very related to the considered problem and makes each proof different from others. These impossibility results identify a necessary topological condition for each problem.

We first interest in the eventual underlying graph computation and prove that this problem does not admit a deterministic solution on any TVG such that its underlying graph has at least one cycle (Section 4.1). Then, we focus on cover problems and prove a necessary topological condition for the Minimal Dominating Set construction and the Maximal Matching construction respectively (Section 4.2 and 4.3). For Minimal Dominating Set, we prove that any deterministic algorithm requires that there exists a set of vertices that is a minimal dominating set for every connected spanning subgraph of the underlying graph. Regarding Maximal Matching, our result states that this problem is impossible to solve in a deterministic way unless the underlying graph of the TVG admits a maximal matching containing only bridges.

Maybe surprisingly, note that all these conditions involve the underlying graph of the TVG whereas the definition of each problem is related to the *eventual* underlying graph. This is intuitively due the fact that to no process is able to determine if, along its adjacent edges, there exists some eventual missing edges or not (in any TVG of $\mathcal{COT}$ whose underlying graph is not a tree). That implies that the algorithm must converge to a solution that satisfies the problem for *any* possible eventual underlying graph of the TVG (*i.e.* any connected spanning subgraph of the underlying graph). Our necessary conditions exhibited in this section follow from the fact that such a solution does not exists for every graph.

4.1 Eventual Underlying Graph Computation

Roughly speaking, the eventual underlying graph computation problem consists in, for each process, to compute *eventually* the set of vertices and of edges of the eventual underlying graph of the TVG. More formally, we can specify this problem in the following way.

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

Unfortunately, this very natural problem is impossible to solve deterministically on $\mathcal{COT}$ apart from trivial cases where the underlying graph is a tree. This latter case is trivial since such TVG does not admit eventual missing edges. This impossibility is captured by the following theorem.

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 $G \oplus \{(e_1, \mathcal{T}_{e_1}), \ldots, (e_k, \mathcal{T}_{e_k})\}$ (with, for any $i \in \{0, \ldots, k\}$, $e_i \in E$ and $\mathcal{T}_{e_i} \subseteq \mathcal{T}$) as the TVG $(V, E, \mathcal{T}, \rho', \zeta, \phi)$ with:

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

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 $\eta_i = +\infty$);
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

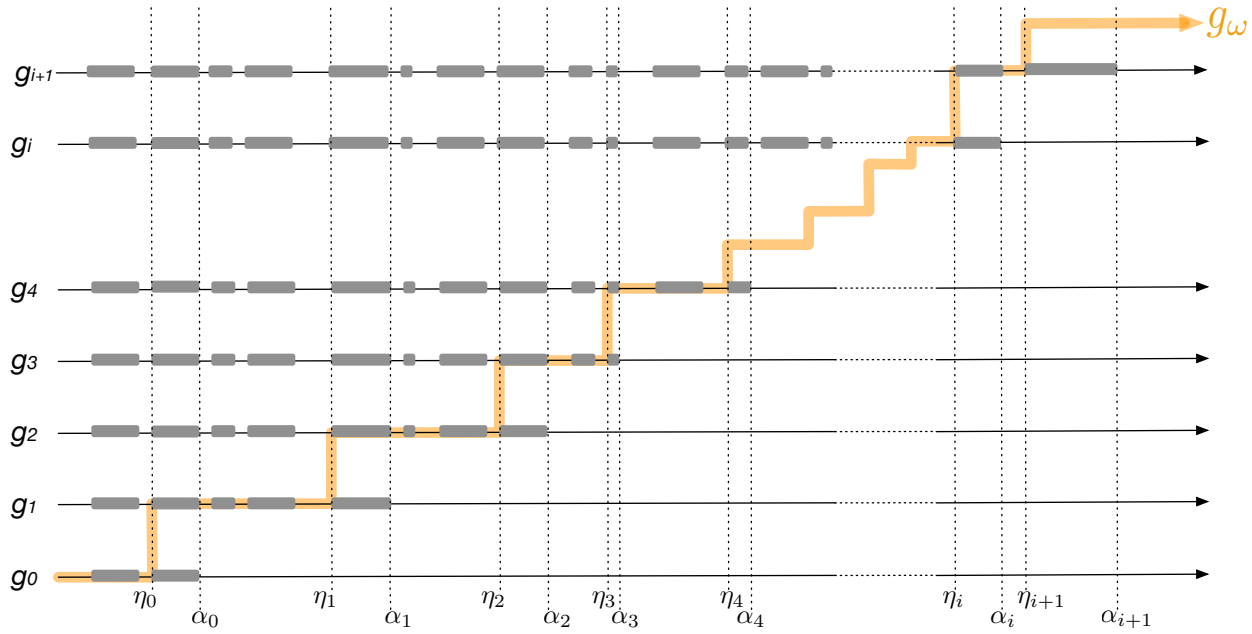

Figure 1: Construction of $(G_n)_{n\in\mathbb{N}}$ in the proof of Theorem 2. Grey bold lines represent instants where e belongs to the graph outputted by all process of V.

$\mathcal{COT}|_{\mathcal{F}}$, 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 $d_\Gamma(G_k, G_\omega) = 2^{-\eta_k}$ 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.

By Theorem 1, 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. □

4.2 Minimal Dominating Set Construction

Recall that, in a static distributed system, a dominating set D is a subset of processes of the system such that each process that does not belong to D have at least one neighbor in D. Such a dominating set is minimal when it has is no strict subset that is also a dominating set.

Regarding dynamic distributed systems, two different approaches have been proposed to handle minimal dominating set problem. We survey them quickly here and show that these definitions seem not relevant in our context, that motivates the need of our new definition presented in this section.

The most natural way to extend minimal dominating set definition in the context of dynamic systems is presented in [11]. In this work, the dynamic graph is seen as a sequence of static graphs and a new minimal dominating set is computed at each topological change. This approach is not suitable in the case of highly dynamic systems since the system may be always in computation phase (the computation of the new dominating set at each topological change is not instantaneous). In this case, the dominating set may be never stable and is then useless for the application that required it.

The second approach, proposed by [4], consists in computing a stable dominating set on the underlying graph of the TVG. This approach is interesting since the outputted dominating set is stable in spite of the dynamics of the system but is still not suitable for our purpose. Indeed, as the dominating set is computed on the underlying graph that may contain eventual missing edges, it is possible for a process to be dominated only through such edges. In other words, a dominated process may have eventually only dominated neighbors, that is counter-intuitive for a minimal dominating set and makes sense only in TVGs where there is no eventual missing edges.

To overcome flaws of precedent definitions in our context of highly dynamic distributed systems (captured by the class of TVGs $\mathcal{COT}$), we propose a third definition in which we require the outputted minimal dominating set to be stable and each dominated process to be infinitely often neighbor of at least one dominating process. In other words, we want to compute a minimal dominating set on the *eventual* underlying graph. Note that this definition is exactly the same as the one of [4] in TVGs where there is no eventual missing edges. We specify the minimal dominating set construction problem over TVGs as follows.

Definition 6 (Minimal dominating set over time) *A set of processes M is a minimal dominating set over time (MDST for short) of a TVG G if M is a minimal dominating set of U^{ω}_{G}.*

Specification 2 (Minimal dominating set) *An algorithm $\mathcal{A}$ satisfies the minimal dominating set specification for a class of TVGs $\mathcal{C}$ if the execution $e = \gamma_0, \gamma_1, \ldots$ of $\mathcal{A}$ on every TVG G of $\mathcal{C}$ has a suffix $e_i = \gamma_i, \gamma_{i+1}, \ldots$ for a given $i \in \mathbb{N}$ such that each process constantly outputs a boolean value in any configuration of e_i and that the set of processes outputting true is a minimal dominating set over time of G.*

In the following, we prove that this problem does not admit a deterministic solution for any TVG of $\mathcal{COT}$. In particular, we prove that any deterministic algorithm requires that there exists a set of vertices that is a minimal dominating set for every connected spanning subgraph of the underlying graph of the TVG. Such a minimal dominating set is hereafter referenced as *robust*–refer to Figure 2 for an illustration. A formal definition follows.

Definition 7 (Robust minimal dominating set) *A robust minimal dominating set of a (static) graph G is a subset of processes of G that is a minimal dominating set of every connected spanning subgraph of G.*

Intuitively, the impossibility comes from the following fact. As no process is able to detect eventual missing edges, the minimal dominated set computed by any algorithm must be a minimal dominated set of any possible eventual underlying graph, that is of any connected subgraph of the underlying graph. In other words, the computed minimal dominated set is a robust minimal dominating set. The existence of such a set is then a necessary condition to the existence of an algorithm to compute a minimal dominating set over time. The main difficulty of the formal proof of this result lies in the construction of the TVGs sequence that allows us to apply Theorem 1.

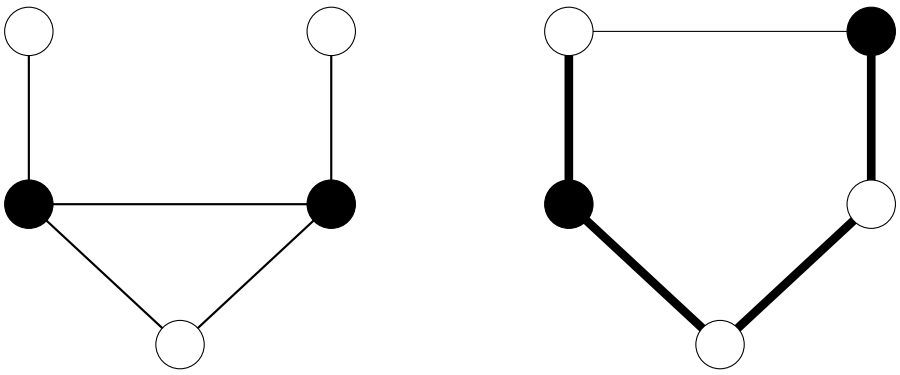

Figure 2: The graph on the left side admits a robust minimal dominating set (the set of black vertices). The graph on the right side admits no robust minimal dominating set (for instance, the black minimal dominating set is not a dominating set of the connected spanning subgraph with bold edges).

Theorem 3 *For any set of (static) graphs $\mathcal{F}$ containing at least one graph that does not admit a robust minimal dominating set, there exists no deterministic algorithm that satisfies the minimal dominating set specification for $\mathcal{COT}|_{\mathcal{F}}$.*

Proof. Let us introduce some notation first. We define, for any TVG $G = (V, E, \mathcal{T}, \rho, \zeta, \phi)$, the TVG $G \odot \{(E_i, \mathcal{T}_i) | i \in I\}$ (with $I \subseteq \mathbb{N}$ and for any $i \in I$, $E_i \subseteq E$ and $\mathcal{T}_i \subseteq \mathcal{T}$) as the TVG $(V, E, \mathcal{T}, \rho', \zeta, \phi)$ with:

$$\rho'(e,t) = \begin{cases} 0 \text{ if } \exists i \in I, e \in E_i \text{ and } t \in \mathcal{T}_i \\ 1 \text{ if } \exists i \in I, e \in E \setminus E_i \text{ and } t \in \mathcal{T}_i \\ \rho(e,t) \text{ otherwise} \end{cases}$$

By contradiction, assume that there exists a set of (static) graphs $\mathcal{F}$ containing at least one graph that does not admit a robust minimal dominating set and that there exists a deterministic algorithm $\mathcal{A}$ that satisfies the minimal dominating set specification for $\mathcal{COT}|_{\mathcal{F}}$. In consequence, any process that executes $\mathcal{A}$ outputs a boolean value at any time.

Let $G = (V, E, \mathcal{T}, \rho, \zeta, \phi)$ be a TVG of $\mathcal{COT}|_{\mathcal{F}}$ such that U_G does not admit a robust minimal dominating set and that there exists $t_0 \in \mathcal{T}$ such that $\forall e \in E, \forall t \leq t_0, \rho(e,t) = 1$ (G exists by construction of $\mathcal{F}$ and by definition of $\mathcal{COT}|_{\mathcal{F}}$). We construct then a sequence $(G_n)_{n \in \mathbb{N}}$ of TVGs as follows. We set $G_0 = G$. Assume that we have already $G_i = (V, E, \mathcal{T}, \rho', \zeta, \phi)$ for a given $i \in \mathbb{N}$ such that $G_i \in \mathcal{COT}|_{\mathcal{F}}$, $U_{G_i} = U_G$, and $\exists \alpha_{i-1} > t_0, \forall e \in E, \forall t > \alpha_i, \rho'(e,t) = \rho(e,t)$. Then, we define inductively G_{i+1} as follows:

1. Consider the execution of $\mathcal{A}$ over G_i and let $\eta_i \in \mathcal{T}$ be the smallest time strictly greater than α_{i-1} from which the set of processes that output true is constant (η_i exists by assumption on $\mathcal{A}$ since $G_i \in \mathcal{COT}|_{\mathcal{F}}$);

2. Let M_i be the minimal dominating set computed by $\mathcal{A}$ on G_i (*i.e.* the set of processes of G_i outputting true after η_i). As $U_{G_i} = U_G$, we know by assumption on U_G that U_{G_i} does not admit a robust minimal dominating set. In particular, M_i is not a robust minimal dominating set of U_{G_i}. Hence, there exists a process p_i of $V \setminus M_i$ such that the set of edges $E_i = \{\{p_i, q\} | q \in M_i \cap \mathcal{N}_{p_i}\}$ is not a cut-set of U_{G_i};

3. Let $G'_i = G_i \odot \{(E_i, \mathcal{T} \cap]\eta_i, +\infty[)\}$.

4. Remark that $U_{G'_i} = U_{G_i} = U_G$ (by construction of G'_i since $\eta_i > t_0$) and that $U^{\omega}_{G'_i}$ is connected (since $E(U^{\omega}_{G'_i}) = E(U_G) \setminus E_i$ by construction[3] and E_i is not a cut-set of U_G). Hence, $G'_i \in \mathcal{COT}|_{\mathcal{F}}$ and we

[3]where $E(G)$ denotes the set of edges of G.

can consider the execution of $\mathcal{A}$ over G'_i. Let $\alpha_i \in \mathcal{T}$ be the smallest time strictly greater than η_i from which the set of processes that output true is constant. Let M'_i be the minimal dominating set computed by $\mathcal{A}$ on G'_i (*i.e.* the set of processes of G'_i outputting true after α_i). Note that $M'_i \neq M_i$ since M_i is not a minimal dominating set of $U^{\omega}_{G'_i}$ (recall that, in $U^{\omega}_{G'_i}$, p_i has no neighbor in M_i);

5. Let $G_{i+1} = G_i \odot \{(E_i, \mathcal{T} \cap]\eta_i, \alpha_i])\}$.

It is straightforward to check that this construction ensures that, if there exists $G_i = (V, E, \mathcal{T}, \rho', \zeta, \phi)$ for a given $i \in \mathbb{N}$ such that $G_i \in \mathcal{COT}|_{\mathcal{F}}$, $U_{G_i} = U_G$, and $\exists \alpha_{i-1} > t_0, \forall e \in E, \forall t > \alpha_i, \rho'(e,t) = \rho(e,t)$, then G_{i+1} satisfies the same property. Moreover, as $G_0 = G$, this property is naturally satisfied for $i = 0$ with any $\alpha_{-1} > t_0$. Hence, the sequence $(G_n)_{n\in\mathbb{N}}$ is well-defined. Note that, for any $i \in \mathbb{N}$, $\eta_i < \alpha_i$ and $\alpha_i < \eta_{i+1}$ (by construction).

That allows us to define the following TVG: $G_\omega = G \odot \{(E_i, \mathcal{T} \cap]\eta_i, \alpha_i])|i \in \mathbb{N}\}$. Note that $U_{G_\omega} = U_G$ and then that G_ω belongs to $\mathcal{COT}|_{\mathcal{F}}$. Observe that, for any $k \in \mathbb{N}^*$, we have $d_\Gamma(G_k, G_\omega) = 2^{-\eta_k}$ 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 ready to apply 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, for any $i \in \mathbb{N}^*$, the set of processes that output true in G_ω at η_i is M_i and the set of processes that output true in G_ω at α_i is M'_i. As we know that $M_i \neq M'_i$ for any $i \in \mathbb{N}$, we obtain that the set of processes that output true in G_ω never converges, that contradicts the fact that $\mathcal{A}$ satisfies the minimal dominating set specification for $\mathcal{COT}|_{\mathcal{F}}$ and ends the proof. □

4.3 Maximal Matching

As the maximal matching shares similarities with other cover problems (such as the minimal dominating set), we can adopt the same approach than in the previous section to specify it in the context of dynamic systems.

Definition 8 (Maximal matching over time) *A set of processes M is a maximal matching over time of a TVG G if M is a maximal matching of U^{ω}_G.*

Specification 3 (Maximal matching) *An algorithm $\mathcal{A}$ satisfies the maximal matching specification for a class of TVGs $\mathcal{C}$ if the execution $e = \gamma_0, \gamma_1, \ldots$ of $\mathcal{A}$ on every TVG G of $\mathcal{C}$ has a suffix $e_i = \gamma_i, \gamma_{i+1}, \ldots$ for a given $i \in \mathbb{N}$ such that each process constantly outputs a boolean value in any configuration of e_i and that the set of processes outputting true is a maximal matching over time of G.*

In this section, we prove that this problem is impossible to solve in a deterministic way for a TVG of the class $\mathcal{COT}$ unless the underlying graph of this TVG admits a maximal matching containing only bridges—refer to Figure 3 for an illustration. Intuitively, this condition follows from the following fact. Such a maximal matching is by construction a maximal matching of any connected spanning subgraph of the underlying graph and hence of any potential eventual underlying graph of the TVG. Then, we obtain the following theorem.

Theorem 4 *For any set of (static) graphs $\mathcal{F}$ containing at least one graph that does not admit a maximal matching containing only bridges, there exists no deterministic algorithm that satisfies the maximal matching specification for $\mathcal{COT}|_{\mathcal{F}}$.*

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

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

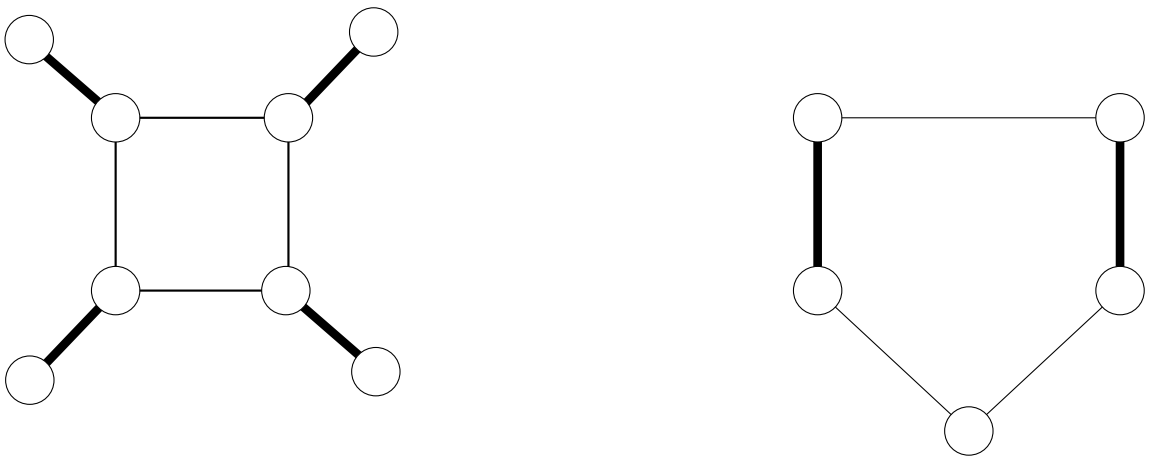

Figure 3: The graph on the left side admits a maximal matching containing only bridges (the set of bold edges). The graph on the right side admits no maximal matching containing only bridges.

By contradiction, assume that there exists a set of (static) graphs $\mathcal{F}$ containing at least one graph that does not admit a maximal matching containing only bridges and that there exists a deterministic algorithm $\mathcal{A}$ that satisfies the maximal matching specification for $\mathcal{COT}|_{\mathcal{F}}$. In consequence, any process that executes $\mathcal{A}$ outputs a boolean value at any time.

Let $G = (V, E, \mathcal{T}, \rho, \zeta, \phi)$ be a TVG of $\mathcal{COT}|_{\mathcal{F}}$ such that U_G does not admit a maximal matching containing only bridges and that $\forall e \in E, \forall t \in \mathcal{T}, \rho(e,t) = 1$ (G exists by construction of $\mathcal{F}$ and by definition of $\mathcal{COT}|_{\mathcal{F}}$). We construct then a sequence $(G_n)_{n\in\mathbb{N}}$ of TVGs as follows. We set $G_0 = G$. Assume that we have already $G_i = (V, E, \mathcal{T}, \rho', \zeta, \phi)$ for a given $i \in \mathbb{N}$ such that $G_i \in \mathcal{COT}|_{\mathcal{F}}$, $U_{G_i} = U_G$, and $\exists \alpha_{i-1} > 0, \forall e \in E, \forall t > \alpha_i, \rho'(e,t) = \rho(e,t)$. Then, we define inductively G_{i+1} as follows:

1. Consider the execution of $\mathcal{A}$ over G_i and let $\eta_i \in \mathcal{T}$ be the smallest time strictly greater than α_{i-1} from which the set of processes that output true is constant (η_i exists by assumption on $\mathcal{A}$ since $G_i \in \mathcal{COT}|_{\mathcal{F}}$);
2. Let M_i be the maximal matching computed by $\mathcal{A}$ on G_i (*i.e.* the set of processes of G_i outputting true after η_i). As $U_{G_i} = U_G$, we know by assumption on U_G that U_{G_i} does not admit a matching containing only bridges. In particular, there exists a non-bridge edge e_i in M_i;
3. Let $G'_i = G_i \ominus \{(e_i, \mathcal{T} \cap]\eta_i, +\infty[)\}$.
4. Remark that $U_{G'_i} = U_{G_i} = U_G$ (by construction of G'_i since $\eta_i > 0$) and that $U^{\omega}_{G'_i}$ is connected (since e_i is not a bridge). Hence, $G'_i \in \mathcal{COT}|_{\mathcal{F}}$ and we can consider the execution of $\mathcal{A}$ over G'_i. Let $\alpha_i \in \mathcal{T}$ be the smallest time strictly greater than η_i from which the set of processes that output true is constant. Let M'_i be the maximal matching computed by $\mathcal{A}$ on G'_i (*i.e.* the set of processes of G'_i outputting true after α_i). Note that $M'_i \neq M_i$ since e_i does not belongs to $U^{\omega}_{G'_i}$ by construction;
5. Let $G_{i+1} = G_i \ominus \{(e_i, \mathcal{T} \cap]\eta_i, \alpha_i])\}$.

It is straightforward to check that this construction ensures that, if there exists $G_i = (V, E, \mathcal{T}, \rho', \zeta, \phi)$ for a given $i \in \mathbb{N}$ such that $G_i \in \mathcal{COT}|_{\mathcal{F}}$, $U_{G_i} = U_G$, and $\exists \alpha_{i-1} > 0, \forall e \in E, \forall t > \alpha_i, \rho'(e,t) = \rho(e,t)$, then G_{i+1} satisfies the same property. Moreover, as $G_0 = G$, this property is naturally satisfied for $i = 0$ with any $\alpha_{-1} > 0$. Hence, the sequence $(G_n)_{n\in\mathbb{N}}$ is well-defined. Note that, for any $i \in \mathbb{N}$, $\eta_i < \alpha_i$ and $\alpha_i < \eta_{i+1}$ (by construction).

That allows us to define the following TVG: $G_\omega = G \ominus \{(e_i, \mathcal{T}\cap]\eta_i, \alpha_i])|i \in \mathbb{N}\}$. Note that $U_{G_\omega} = U_G$ and then that G_ω belongs to $\mathcal{COT}|_\mathcal{F}$. Observe that, for any $k \in \mathbb{N}^*$, we have $d_\Gamma(G_k, G_\omega) = 2^{-\eta_k}$ 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 ready to apply 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, for any $i \in \mathbb{N}^*$, the set of processes that output true in G_ω at η_i is M_i and the set of processes that output true in G_ω at α_i is M'_i. As we know that $M_i \neq M'_i$ for any $i \in \mathbb{N}$, we obtain that the set of processes that output true in G_ω never converges, that contradicts the fact that $\mathcal{A}$ satisfies the maximal matching specification for $\mathcal{COT}|_\mathcal{F}$ and ends the proof. □

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 then used the above result to prove several impossibility results related to long-lived time-varying graphs with minimal assumptions to show the effectiveness of our framework.

Namely, we proved first that it is impossible to compute deterministically the eventual underlying graph (*i.e.* the set of edges that appear infinitely often) thenceforth the underlying graph is not a tree. Second, we proved that there exists no deterministic algorithm to compute a minimal dominating set on the time-varying graph if the underlying graph of this latter does not admit a robust minimal dominating set (defined as a minimal dominating set that have the property to be a minimal dominating of any spanning connected subgraph of the underlying graph). Finally, we showed the impossibility of deterministically build a maximal matching of a time-varying graph unless its underlying graph admits a maximal matching containing only bridges.

Note that we provide in this paper the proof that these conditions are necessary to the existence of a solution. The sufficiency of these conditions is not studied in the context of this work but is derivable from algorithms proposed in [7]. This work opens some research perspectives. It would be interesting to characterize the set of topologies that corresponds to the conditions identified as necessary in this problem for both the minimal dominating set problem and maximal matching problem. It would be also attractive to study whether our general result can be applied to other problems, including non-cover problems.

References

[1] Aris Anagnostopoulos, Ravi Kumar, Mohammad Mahdian, Eli Upfal, and Fabio Vandin. Algorithms on evolving graphs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS'12)*, pages 149–160. ACM, 2012.

[2] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing (PODC 1984)*, pages 278–281, 1984.

[3] Nicolas Braud Santoni, Swan Dubois, Mohamed Hamza Kaaouachi, and Franck Petit. A generic framework for impossibility results in time-varying graphs. In *IEEE 29nd International Symposium on Parallel and Distributed Processing (IPDPS 2015), 17th Workshop on Advances in Parallel and Distributed Computational Models (APDCM 2015)*, pages 483–489. IEEE, 2015.

[4] A. Casteigts and P. Flocchini. Deterministic algorithms in dynamic networks: Problems, analysis, and algorithmic tools. Technical report, Defence Research and Development Canada, 2013-020, 2013.

[5] 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.

[6] A. Casteigts, B. Mans, and L. Mathieson. On the feasibility of maintenance algorithms in dynamic graphs. Technical report, arXiv – abs/1107.2722, 2011.

[7] Swan Dubois, Mohamed Hamza Kaaouachi, and Franck Petit. Dominating set in highly dynamic distributed systems. In *17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2015)*, volume to appear of *Lecture Notes in Computer Science (LNCS)*, Edmonton, Canada, 2015. Springer Berlin / Heidelberg.

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

[9] J. Schneider and R. Wattenhofer. Coloring unstructured wireless multi-hop networks. In *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC 2009)*, pages 210–219, 2009.

[10] J. Schneider and R. Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5-6):349–361, 2010.

[11] J. Whitbeck, M. Dias de Amorim, V. Conan, and J.-L. Guillaume. Temporal reachability graphs. In *The 18th Annual International Conference on Mobile Computing and Networking (MobiCom'12)*, pages 377–388, 2012.

[12] 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.