On the Expressivity of Time-Varying Graphs

Arnaud Casteigts[a], Paola Flocchini[b], Emmanuel Godard[c], Nicola Santoro[d], Masafumi Yamashita[e]

[a] *LaBRI, University of Bordeaux, France*
[b] *SEECS, University of Ottawa, Canada*
[c] *LIF, Université Aix-Marseille, France*
[d] *SCS, Carleton University, Ottawa, Canada*
[e] *Kyushu University, Fukuoka, Japan*

Abstract

In highly dynamic systems (such as wireless mobile ad-hoc networks, robotic swarms, vehicular networks, etc.) connectivity does not necessarily hold at a given time but temporal paths, or *journeys*, may still exist over time and space, rendering computing possible; some of these systems allow *waiting* (i.e., pauses at intermediate nodes, also referred to as store-carry-forward strategies) while others do not. These systems are naturally modelled as *time-varying graphs*, where the presence of an edge and its latency vary as a function of time; in these graphs, the distinction between waiting and not waiting corresponds to the one between indirect and direct journeys.

We consider the *expressivity* of time-varying graphs, in terms of the languages generated by the feasible journeys. We examine the impact of waiting by studying the difference in the type of language expressed by indirect journeys (i.e., waiting is allowed) and by direct journeys (i.e., waiting is unfeasible), under various assumptions on the functions that control the presence and latency of edges. We prove a general result which implies that, if *waiting is not allowed*, then the set of languages $\mathcal{L}_{nowait}$ that can be generated contains all computable languages when the presence and latency functions are computable. On the other end, we prove that, if *waiting is allowed*, then the set of languages $\mathcal{L}_{wait}$ contains all and only regular languages; this result, established using algebraic properties of quasi-orders, holds even if the presence and latency are unrestricted (e.g., possibly non-computable) functions of time.

In other words, we prove that, when waiting is allowed, the power of the accepting automaton can drop drastically from being at least as powerful as a Turing machine, to becoming that of a Finite-State Machine. This large gap provides an insight on the impact of waiting in time-varying graphs.

We also study *bounded waiting*, in which waiting is allowed at a node for at most d time units, and prove that $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$; that is, the power of the accepting automaton decreases only if waiting time is unbounded.

1. Introduction

1.1. Highly Dynamic Networks and Time-Varying Graphs

The study of *highly dynamic networks* focuses on networked systems where changes in the topology are extensive, possibly unbounded, and occur continuously; in particular, connectivity might never be present. For example, in wireless mobile ad hoc networks, the topology depends on the current distance between *mobile* nodes: an edge exists between them at a given time if they are within communication range at that time. Hence, the topology changes continuously as the movements of the entities destroy old connections and create new ones. These changes can be dramatic; connectivity does not necessarily hold, at least with the usual meaning of contemporaneous end-to-end multi-hop paths between any pair of nodes, and the network may actually be disconnected at every time instant. These infrastructure-less highly dynamic networks, variously called *delay-tolerant*, *disruptive-tolerant*, *challenged*, *epidemic*, *opportunistic*, have been long and extensively investigated by the engineering community and, more recently, by distributed computing researchers (*e.g.* [38, 44, 47, 51]). Some of these systems provide the entities with store-carry-forward-like mechanisms (e.g., local buffering) while others do not. In presence of local buffering, an entity wanting to communicate with a specific other entity, can wait until the opportunity of communication presents itself; clearly, if such buffering mechanisms are not provided, waiting is not possible.

These highly dynamic networks are modelled in a natural way as *time-varying graphs* or *evolving graphs* (e.g., [18, 27]). In a time-varying graph (TVG), edges between nodes exist only at certain times (in general, unknown to the nodes themselves) specified by a *presence* function. Another component of TVGs is the *latency* function, which indicates the time it takes to cross a given edge at a given time. The lifetime of a TVG can be arbitrary, that is time could be discrete or continuous, and the presence and latency functions can vary from finite automata to Turing computable functions and even non-computable functions.

A crucial aspect of time-varying graphs is that a path from a node to another might still exist over time, even though at no time the path exists in its entirety; it is this fact that renders computing possible. Indeed, the notion of "path over time", formally called *journey*, is a fundamental concept and plays a central role in the definition of almost all concepts related to connectivity in time-varying graphs. Examined extensively, under a variety of names (e.g., temporal path, schedule-conforming path, time-respecting path, trail), informally a journey is a walk[1] $<e_1, e_2, ..., e_k>$ with a sequence of time instants $<t_1, t_2, ..., t_k>$ where edge e_i exists at time t_i and its latency ζ_i at that time is such that $t_{i+1} \geq t_i + \zeta_i$.

The distinction between absence and availability of local buffering in highly dynamic systems corresponds in time-varying graphs to the distinction between

[1]A walk is a path with possibly repeated edges.

a journey where $\forall i, t_{i+1} = t_i + \zeta_i$ (a *direct* journey), and one where it may happen that, for some i, $t_{i+1} > t_i + \zeta_i$ (an *indirect* journey).

In this paper, we are interested in studying the difference between direct and indirect journeys, that is the difference that the possibility of waiting creates in time-varying graphs.

1.2. Main Contributions

In a time-varying graph $\mathcal{G}$, a journey can be viewed as a word on the alphabet of the edge labels; in this light, the class of feasible journeys in $\mathcal{G}$ defines a language $L_f(\mathcal{G})$ expressed by $\mathcal{G}$, where $f \in \{wait, nowait\}$ indicates whether or not indirect journeys are allowed. In this paper we examine the complexity of time-varying graphs in terms of their *expressivity*, that is of the language defined by the journeys, and establish results showing the difference that the possibility of waiting creates.

We will investigate and demonstrate the varying expressivity we get in the *non-waiting* case and the constant expressivity we get in the *waiting* case.

Given a class of functions Φ, we consider the class $\mathcal{U}_\Phi$ of TVGs whose presence and latency functions belong to Φ. More precisely, we focus on the sets of languages $\mathcal{L}^\Phi_{nowait} = \{L_{nowait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}_\Phi\}$ and $\mathcal{L}^\Phi_{wait} = \{L_{wait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}_\Phi\}$ expressed when waiting is, or is not allowed. For each of these two sets, the complexity of recognizing any language in the set (that is, the computational power needed by the accepting automaton) defines the complexity of the environment.

We first study the expressivity of time-varying graphs when waiting is not allowed, that is the only feasible journeys are direct ones. We show that, for any computable language L, there exists a time-varying graph $\mathcal{G}$, with computable functions for presence and latency, such that $L_{nowait}(\mathcal{G}) = L$. We actually prove the stronger result that, given a class of functions Φ, the set $\mathcal{L}^\Phi_{nowait}$ contains the languages recognizable by Φ.

We next examine the expressivity of time-varying graphs if indirect journeys are allowed. We prove that, for any class Φ, $\mathcal{L}^\Phi_{wait}$ is precisely the set of *regular* languages; even if the presence and latency functions are arbitrarily complex (e.g., non-computable) functions of time, only regular languages can be generated. The proof is algebraic and based on order techniques, relying on a theorem by Harju and Ilie [34] that enables to characterize regularity from the closure of the sets from a well quasi-order. In other words, we prove as a main corollary that, when waiting is allowed, the power of the accepting automaton drops drastically from being (possibly) as powerful as a Turing Machine, to becoming that of a Finite-State Machine.

To better understand the impact of waiting on the expressivity of time-varying graphs, we then turn our attention to *bounded waiting*; that is when indirect journeys are considered feasible if the pause between consecutive edges in the journeys has a duration bounded by $d > 0$. At each step of the journey, waiting is allowed only for at most d time units. Hence, we examine the set $\mathcal{L}_{wait[d]}$ of the languages expressed by time-varying graphs when waiting is allowed up to d time units. In fact, we prove that for any fixed $d \geq 0$,

$\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$, which implies that the expressivity of time-varying graphs is not impacted by allowing waiting for a limited amount of time.

1.3. Related Work

The literature on dynamic networks and dynamic graphs could fill a volume. Here we briefly mention only some of the work most directly connected to the results of this paper. In this light, noticeable is the pioneering work, in distributed computing, by Awerbuch and Even on broadcasting in dynamic networks [6], and, in graph theory, by Harari and Gupta on models of dynamic graphs [33].

The idea of representing a dynamic graph as a sequence of (static) graphs, called *evolving graph* (EG), was formalized in [27] to study basic dynamic network problems initially from a centralized point of view [8, 13]. In an evolving graph representation, the dynamics of the system is viewed as a sequence of *global* snapshots (taken either in discrete steps or when events occur). This notion has been subsequently re-discovered by researchers who, unaware of the pre-existing literature, have called it with different names; in particular, the term "time-varying graph" was first used in such a context [48].

The notion of *time-varying graph* (TVG) used here has been introduced in [18]. It is theoretically more general than that of evolving graph; the two notions are computationally equivalent in the case of countable events (edge appearence/disappearance). In a time-varying graph representation, the dynamics of the system is expressed in terms of the changes in the *local* viewpoint of the entities.

Both EG and TVG have been extensively employed in the analysis of basic problems such as routing, broadcasting, gossiping and other forms of information spreading (*e.g.*, [5, 9, 17, 21, 25, 29, 47, 49, 50]); to study problems of exploration (*e.g.* [1, 12, 28, 29, 30, 36, 37]); to examine fault-tolerance, consensus and security (*e.g.*, [11, 22, 31, 42, 43]); for investigating leader election, counting and computing network information (*e.g.*, [4, 16, 24, 32]); to examine computability issues (*e.g.*, [15, 45]); for studying the probabilistic analysis of informations spreading and use of randomizationn (*e.g.* [7, 19, 20, 23]); to identify graph components with special properties (e.g., [3, 40]); and to investigate emerging properties in social networks (*e.g.*, [10, 14, 39, 41, 48]).

A characterization of classes of TVGs with respect to properties typically assumed in distributed computing research can be found in [18]. The impact of bounded waiting in dynamic networks has been investigated for exploration [37].

The closest concept to TVG-automata, defined in this paper, are the well-established *Timed Automata* proposed by [2] to model real-time systems. A timed automaton has real valued clocks and the transitions are guarded with finite comparisons on the clock values; with only one clock and no reset it is a TVG-automaton with 0 latency. Note that, even in the simple setting of timed automata, some key problems, like inclusion, are undecidable for timed languages in the non-deterministic case, while the deterministic case lacks some expressive power. Further note that we focus here on the properties of the un-timed part of the journeys (i.e. the underlying walk made of the edges that are crossed), and given that the guards (presence and latency) can be arbitrary functions, the reachability problem is obviously not decidable for TVG-automaton. This is probably what explains that, to the best of our knowledge, such systems have not been considered for these classical questions. We are here mainly interested in comparing the expressivity of waiting and non-waiting in TVGs, which is a more unusual question.

2. Definitions and Terminology

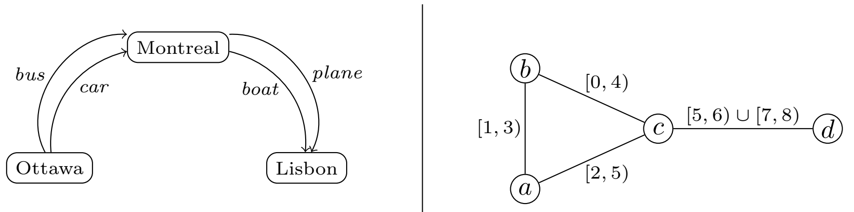
2.1. Time-varying graphs

Following [18], we define a *time-varying graph* (*TVG*) as a quintuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where V is a finite set of entities or *nodes*; $E \subseteq V \times V \times \Sigma$ is a finite set of relations, or *edges*, between these entities, possibly labeled by symbols in an alphabet Σ. The system is studied over a given time span $\mathcal{T} \subseteq \mathbb{T}$ called *lifetime*, where $\mathbb{T}$ is an arbitrary temporal domain, that is, time could be discrete (e.g., $\mathbb{T} = \mathbb{N}$) or continuous (e.g., $\mathbb{T} = \mathbb{R}^+$); $\rho : E \times \mathcal{T} \to \{0, 1\}$ is the edge *presence* function, which indicates whether a given edge is available at a given time; $\zeta : E \times \mathcal{T} \to \mathbb{T}$, is the *latency* function, which indicates the time it takes to cross a given edge if starting at a given date (the latency of an edge could vary in time). In general, both presence and latency are arbitrary functions of the time. The impact of restricting the computability class of presence and latency is further discussed later. In this paper we restrict ourselves to *deterministic* functions.

The directed edge-labeled graph $G = (V, E)$, called the *footprint* of $\mathcal{G}$, may contain loops, and it may have more than one edge between the same nodes, but all with different labels.

Definition 2.1. *A* journey *is a finite sequence* $\langle (e_1, t_1), (e_2, t_2), ..., e_k, t_k) \rangle$ *where* $\langle e_1, e_2, \ldots, e_k \rangle$ *is a walk in the footprint* G, $\rho(e_i, t_i) = 1$ *(for* $1 \leq i < k$*), and* $\zeta(e_i, t_i)$ *is such that* $t_{i+1} \geq t_i + \zeta(e_i, t_i)$ *(for* $1 \leq i < k$*). If* $\forall i, t_{i+1} = t_i + \zeta(e_i, t_i)$ *the journey is said to be* direct*, otherwise* indirect*. We denote by* $\mathcal{J}^*(\mathcal{G})$ *the set of all possible journeys in* $\mathcal{G}$*.*

Time-varying graph introduced in [18], can arguably describe a multitude of different scenarios, from transportation networks to communication networks, complex systems, or social networks. Figure 1 shows two simple examples of TVGs, depicting respectively a transportation network (Figure 1a) and a communication network (Figure 1b). In the transportation network, an edge from node u to node v represents the possibility for some agent to move from u to v; typical edges in this scenario are available on a *punctual* basis, *i.e.,* the presence function ρ for these edges returns 1 only at particular date(s) when the trip can be started. The latency function ζ may also vary from one edge to another, as well as for different availability dates of a same given edge (e.g. variable traffic on the road, depending on the departure time). In



(a) Example of multi-TVG

(b) TVG with presence function depicted

Figure 1: Two examples of time-varying graphs, highlighting (a) the labels, and (b) the presence function.

the communication network of Figure 1b, the labels are not indicated; shown instead are the intervals of time when the presence function ρ is 1. Assuming $\zeta = 1$ for all edges at all times, examples of indirect journeys include $\mathcal{J}_1 = \{(ac, 2), (cd, 5)\}$, and $\mathcal{J}_2 = \{(ab, 2), (bc, 3), (cd, 5)\}$; an example of direct journey is $\mathcal{J}_3 = \{(ab, 2), (bc, 3)\}$; note that $\mathcal{J}_2$ is not a direct journey.

2.2. TVG-automata

Definition 2.2 (TVG-automaton)**.** *Given a time-varying graph $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ whose edges are labeled over Σ, we define a TVG-automaton $\mathcal{A}(\mathcal{G})$ as the 5-tuple $\mathcal{A}(\mathcal{G}) = (\Sigma, S, I, \mathcal{E}, F)$ where*

- Σ *is the input* alphabet*;*
- $S = V$ *is the set of* states*;*
- $I \subseteq S$ *is the set of* initial states*;*
- $F \subseteq S$ *is the set of* accepting states*; and*
- $\mathcal{E} \subseteq S \times \mathcal{T} \times \Sigma \times S \times \mathcal{T}$ *is the set of* transitions *such that* $(s, t, a, s', t') \in \mathcal{E}$ *iff* $\exists e = (s, s', a) \in E : \rho(e, t) = 1, \zeta(e, t) = t' - t$.

In the following we shall denote $(s, t, a, s', t') \in \mathcal{E}$ also by $s, t \overset{a}{\to} s', t'$. A TVG-automaton $\mathcal{A}(\mathcal{G})$ is *deterministic* if for any time $t \in \mathcal{T}$, any state $s \in S$, and any symbol $a \in \Sigma$, there is at most one transition of the form $(s, t \overset{a}{\to} s', t')$; it is *non-deterministic* otherwise.

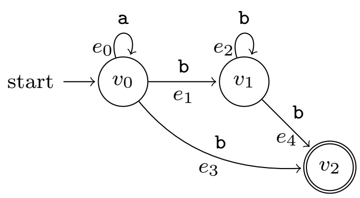
The concept of journey can be extended in a natural way to the framework of TVG-automata.

Definition 2.3 (Journey in a TVG-automaton)**.** *A* journey $\mathcal{J}$ *in a TVG-automaton $\mathcal{A}(\mathcal{G})$ is a finite sequence of transitions*

$$\mathcal{J} = (s_0, t_0 \overset{a_0}{\to} s_1, t_1), (s_1, t'_1 \overset{a_1}{\to} s_2, t_2) \dots (s_{p-1}, t'_{p-1} \overset{a_{p-1}}{\to} s_p, t_p)$$

such that the sequence $\langle (e_0, t_0), (e_1, t'_1), \dots, (e_{p-1}, t'_{p-1}) \rangle$ is a journey in $\mathcal{G}$.

Observe that we have $t_i = t'_{i-1} + \zeta(e_{i-1}, t'_{i-1})$, where $e_i = (s_i, s_{i+1}, a_i)$ (for $0 \le i < p$). Also note that the transitions defining journeys are guarded by arbitrary functions of time.



(a) Structure of $\mathcal{G}_1$

e	$\rho(e,t) = 1$ iff	$\zeta(e,t) =$
e_0	always true	$(p-1)t$
e_1	$t > p$	$(q-1)t$
e_2	$t \neq p^i q^{i-1}$, $i > 1$	$(q-1)t$
e_3	$t = p$	any
e_4	$t = p^i q^{i-1}$, $i > 1$	any

(b) Presence and Latency functions for $\mathcal{G}_1$

Figure 2: A TVG-automaton $\mathcal{G}_1$ such that $L_{nowait}(\mathcal{G}_1) = \{a^n b^n : n \geq 1\}$.

Consistently with the above definitions, we say that $\mathcal{J}$ is *direct* if $\forall i, t'_i = t_i$ (there is no pause between transitions), and *indirect* otherwise.We denote by $\lambda(\mathcal{J})$ the associated word $a_0, a_1, ...a_{p-1}$ and by $start(\mathcal{J})$ and $arrival(\mathcal{J})$ the dates t_0 and t_p, respectively. To complete the definition, an *empty* journey $\mathcal{J}_\emptyset$ consists of a single state, involves no transitions, its associated word is the empty word $\lambda(\mathcal{J}_\emptyset) = \varepsilon$, and its arrival date is the starting date. A journey is said *accepting* if it starts at time $t = 0$ in an initial state $s_0 \in I$ and ends in an accepting state $s_p \in F$ some time later. A TVG-automaton $\mathcal{A}(\mathcal{G})$ *accepts* a word $w \in \Sigma^*$ iff there exists an accepting journey $\mathcal{J}$ such that $\lambda(\mathcal{J}) = w$.

Let $L_{nowait}(\mathcal{G})$ denote the set of words (i.e., the *language*) accepted by TVG-automaton $\mathcal{A}(\mathcal{G})$ using only direct journeys, and let $L_{wait}(\mathcal{G})$ be the language recognized if journeys are allowed to be indirect. Given the set $\mathcal{U}$ of all possible TVGs, let us denote as $\mathcal{L}_{nowait} = \{L_{nowait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ and $\mathcal{L}_{wait} = \{L_{wait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ the sets of all languages being possibly accepted by a TVG-automaton if journeys are constrained to be direct (i.e., no waiting is allowed) and if they are unconstrained (i.e., waiting is allowed), respectively.

In the following, when no ambiguity arises, we will use interchangeably the terms node and state, and the terms edge and transition; the term journey will be used in reference to both TVGs and TVG-automata.

2.3. Example of TVG-automaton

Consider the graph $G = (V, E)$ composed of three nodes: $V = \{v_0, v_1, v_2\}$, and five edges $E = \{e_0 = (v_0, v_0, a),\ e_1 = (v_0, v_1, b),\ e_2 = (v_1, v_1, b),\ e_3 = (v_0, v_2, b),\ e_4 = (v_1, v_2, b))\}$. We show below how to define presence and latency functions, and hence a TVG $\mathcal{G}_1 = (V, E, \mathcal{T}, \rho, \zeta)$, such that, based on direct journeys, the deterministic TVG-automaton $\mathcal{A}(\mathcal{G}_1)$ recognizes the context-free language $\{a^n b^n, n \geq 1\}$.

Consider the automaton $\mathcal{A}(\mathcal{G}_1)$, depicted on Figure 2a, where v_0 is the initial state and v_2 is the accepting state. For clarity, let us assume that $\mathcal{A}(\mathcal{G}_1)$ starts at time 1 (the same behavior could be obtained by modifying slightly the formulas involving t in Table 2b). The presence and latency functions are as shown in Table 2b, where p and q are two distinct prime numbers greater than 1.

It is clear that the a^n portion of the word a^nb^n is read entirely at v_0 within $t = p^n$ time. If $n = 1$, at this time the only available edge is e_3 (labeled b), which allows to correctly accept ab. Otherwise ($n > 1$) at time $t = p^n$, the only available edge is e_1, which allows to start reading the b^n portion of the word. By construction of ρ and ζ, edge e_2 is always present except for the very last b, which has to be read at time $t = p^n q^{n-1}$. At that time, only e_4 is present and the word is correctly recognized. It is easy to verify that only these words are recognized, and the automaton is deterministic. The reader may have noticed the basic principle employed here (and later in the paper) of using latencies as a means to *encode* words into time, and presences as a means to *select* through opening the appropriate edges at the appropriate time.

2.4. Restrictions of Computability.

When considering general TVG-automata, we will investigate whether the class of computability to which the presence and latency functions belong impacts the class of recognizable language by a general TVG-automaton.

Consider a finite alphabet Σ. Let $q = |\Sigma|$ be the size of the alphabet, and w.l.o.g assume that $\Sigma = \{0, \ldots, q-1\}$. Let Φ be a class of functions over the set of integers represented in base q with a *little-endian* encoding (i.e., least significant digit first). For any integer n, $|n|$ denotes the size of the encoding of n in base q.

A function ψ is $\Phi-$computable if $\psi \in \Phi$. A language L is $\Phi-$recognizable if there exists $c \in \mathbb{N}$, $\psi \in \Phi$ such that $L = \psi^{-1}(c)$. By extension, a characteristic function χ_L for a set L is said to be $\Phi-$computable if L is $\Phi-$recognizable.

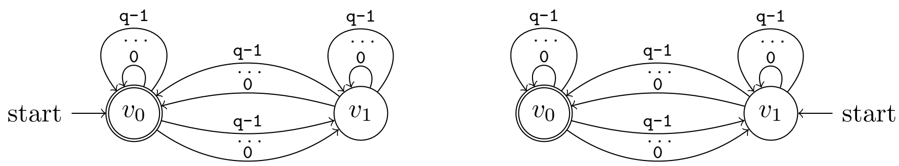
Let L be an arbitrary $\Phi-$computable language defined over the finite alphabet Σ. Let ε denote the empty word; note that L might or might not contain ε. The notation $\alpha.\beta$ indicates the concatenation of $\alpha \in \Sigma^*$ with $\beta \in \Sigma^*$.

Definition 2.4. *A class Φ of functions is $q-$*stable*, for some base q, if it is stable by composition and for any function $\varphi \in \Phi$, for any $p \in \Sigma$,*

1. *the function $\varphi_p : n \mapsto \varphi(n + p \times q^{|n|})$ is in Φ.*
2. *the function $w \mapsto \varphi_p(w) - \varphi(w)$ is in Φ.*

Remark. It should be obvious that standard computability classes satisfy these conditions. For instance, consider finite state transducers with alphabet Σ, adding $p \times q^{|n|}$ to $n \in \mathbb{N}$ can be done with a finite state transducer. Indeed, by assuming *little-endian* encoding in base q for integers in $\mathbb{N}$, such an arithmetic operation corresponds to a concatenation of the letter p at the end. Similarly, for any φ that corresponds to a finite transducer, computing the difference in 2 can be obtained by a finite transducer that outputs 0 for any letter of (the encoding of) n and terminates with a p.

Definition 2.5. *A $\Phi-$TVG-automaton is a TVG-automaton whose presence and latency functions are $\Phi-$computable. The set $\mathcal{L}^{\Phi}_{nowait}$ is the set of languages that can be recognized by a $\Phi-$TVG with no waiting allowed. The set $\mathcal{L}^{\Phi}_{wait}$ is the set of languages that can be recognized by a $\Phi-$TVG with waiting allowed.*



(a) Case with $\varepsilon \in L$ (b) Case with $\varepsilon \notin L$

Figure 3: The TVG $\mathcal{G}_2(L)$ that recognizes the arbitrary computable language L.

3. No Waiting Allowed

This section focuses on the expressivity of time-varying graphs when only *direct* journeys are allowed. We prove that, in this case, the computability class of the presence and latency functions translate directly in the computability class of recognized languages. In other words, for any class Φ, the set $\mathcal{L}^{\Phi}_{nowait}$ of languages recognized by $\Phi-$TVG is at least the set of $\Phi-$recognizable languages. This inclusion is *tight* in the case of classical (Turing) computable function: the set of recognizable languages is exactly the set of recursive languages.

Theorem 3.1. *Let Φ be a $q-$stable class of integer functions. The set $\mathcal{L}^{\Phi}_{nowait}$ of languages recognized by a $\Phi-TVG$ contains the set of $\Phi-$recognizable languages.*

Proof. Consider a class Φ of functions, that is $q-$stable. Consider L a $\Phi-$recognizable language. Denote $\psi \in \Phi$ and $c \in \mathbb{N}$ such that $L = \psi^{-1}(c)$.

Given $p \in \Sigma$, we denote by ψ_p the function of Φ such that $\psi_p : n \mapsto \varphi(n + p \times q^{|n|})$. Note that ψ_p is also in Φ.

Consider now the TVG $\mathcal{G}_2$ where $V = \{v_0, v_1\}$, $E = \{\{(v_0, v_0, i), i \in \Sigma\} \cup \{(v_0, v_1, i), i \in \Sigma\} \cup \{(v_1, v_0, i), i \in \Sigma\} \cup \{(v_1, v_1, i), i \in \Sigma\}\}$. The presence and latency functions are defined relative to which node is the end-point of an edge. For all $u \in \{v_0, v_1\}$, $i \in \Sigma$, and $t \geq 0$, we define

- $\rho((u, v_0, i), t) = true$ if $\psi_i(t) = c$
- $\zeta((u, v_0, i), t) = \psi_i(t) - \psi(t)$
- $\rho((u, v_1, i), t) = true$ if $\psi_i(t) \neq c$
- $\zeta((u, v_1, i), t) = \psi_i(t) - \psi(t)$

Consider the corresponding TVG-automaton $\mathcal{A}(\mathcal{G}_2(L))$ where the unique accepting state is v_0 and the initial state is either v_0 (if $\varepsilon \in L$, see Figure 3a), or v_1 (if $\varepsilon \notin L$ see Figure 3b).

Claim 3.2. *$\mathcal{G}_2(L)$ is a $\Phi-TVG$-automaton. $L_{nowait}(\mathcal{G}_2(L)) = L$.*

Proof. Since Φ is $q-$stable, $\mathcal{G}_2(L)$ presence and latency functions are obviously $\Phi-$computable.

Now, we want to show there is a unique accepting journey $\mathcal{J}$ with $\lambda(\mathcal{J}) = w$ if and only if $w \in L$. We first show that for all words $w \in \Sigma^*$, there is

exactly one direct journey $\mathcal{J}$ in $\mathcal{A}(\mathcal{G}_2(L))$ such that $\lambda(\mathcal{J}) = w$, and in this case $arrival(\mathcal{J}) = \psi(w)$. This is proven by induction on $k \in \mathbb{N}$, the length of the words. It clearly holds for $k = 0$ since the only word of that length is ε and $\psi(\varepsilon) = 0$ (by convention, see above). Let $k \in \mathbb{N}$. Suppose now that for all $w \in \Sigma^*, |w| = k$ we have exactly one associated direct journey, and $arrival(\mathcal{J}) = \psi(w)$.
Consider $w_1 \in \Sigma^*$ with $|w_1| = k + 1$. Without loss of generality, let $w_1 = w.i$ where $w \in \Sigma^*$ and $i \in \Sigma$. By induction there is exactly one direct journey $\mathcal{J}$ with $\lambda(\mathcal{J}) = w$. Let $u = arrival(\mathcal{J})$ be the node of arrival and t the arrival time. By induction, $t \in \psi(\Sigma^*)$; furthermore since the presence function depends only on the node of arrival and not on the node of origin, there exists exactly one transition, labeled i from u. So there exists only one direct journey labeled by w_1. By definition of the latency function, its arrival time is $\psi(w) + (\psi(w.i) - \psi(w)) = \psi_i(w)$. This ends the induction.
We now show that such a unique journey is accepting if and only if $w \in L$. In fact, by construction of the presence function, every journey that corresponds to $w \in L, w \neq \varepsilon$, ends in v_0, which is an accepting state. By construction, the empty journey corresponding to ε ends in the accepting state v_0 if and only if $\varepsilon \in L$. □

For any $\Phi-$recognizable language L, there exists a $\Phi-$TVG-automaton that recognizes L. This concludes the proof of the theorem. □

As a corollary we have

Corollary 3.3. *Let* Turing *be the class of Turing computable integers functions. We have* $\mathcal{L}_{nowait}^{\text{Turing}} = \text{Turing}$

4. Waiting Allowed

We now turn the attention to the case of time-varying graphs where *indirect* journeys are possible. In striking contrast with the non-waiting case, we show that the languages $\mathcal{L}_{wait}^{\Phi}$ recognized by $\Phi-$TVG-automata consists only of regular languages, even if Φ strictly contains the Turing computable functions. Let $\mathcal{R}$ denote the set of regular languages.

Lemma 4.1. *Let Φ be any class of functions containing the constant functions. Then $\mathcal{R} \subseteq \mathcal{L}_{wait}^{\Phi}$.*

Proof. It follows easily from observing that any finite-state machine (FSM) is a particular TVG-automaton whose edges are always present and have a nil latency. The fact that we allow waiting here does not modify the behavior of the automata as long as we consider deterministic FSMs only (which is sufficient), since at most one choice exists at each state for each symbol read. By considering exactly the same initial and final states, for any regular language L, we get a corresponding TVG $\mathcal{G}$ such that $L_{wait}(\mathcal{G}) = L$. □

The reverse inclusion is more involved. Consider a TVG-automaton $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ with labels in Σ and with arbitrary ρ and ζ, we have to show that $L_{wait}(\mathcal{G}) \in \mathcal{R}$.

The proof is algebraic, and based on order techniques, relying on a theorem of Harju and Ilie (Theorem 6.3 in [34]) that enables to characterize regularity from the closure of the sets from a well quasi-order. We will use here an inclusion order on journeys (to be defined formally below). Informally, a journey $\mathcal{J}$ is included in another journey $\mathcal{J}'$ if its sequence of transitions is included (in the same order) in the sequence of transitions of $\mathcal{J}'$. It should be noted that sets of indirect journeys from one node to another are obviously closed under this inclusion order (on the journey $\mathcal{J}$ it is possible to wait on a node as if the missing transitions from $\mathcal{J}'$ were taking place), which is not the case for direct journeys as it is not possible to wait. In order to apply the theorem, we have to show that this inclusion order is a well quasi-order, i.e. that it is not possible to find an infinite set of journeys such that none of them could be included in another from the same set.

Let us first introduce some definitions and results about quasi-orders. We denote by $\leq$ a quasi-order over a given set Q (this is simply a reflexive and transitive relation). A set $X \subset Q$ is an *antichain* if all elements of X are pairwise incomparable. The quasi-order $\leq$ is *well founded* if in Q, there is no infinite descending sequence $x_1 \geq x_2 \geq x_3 \geq \dots$ (where $\geq$ is the inverse of $\leq$) such that for no i, $x_i \leq x_{i+1}$. If $\leq$ is well founded and all antichains are finite then $\leq$ is a *well quasi-order* on Q. When $Q = \Sigma^*$ for alphabet Σ, a quasi-order is *monotone* if for all $x, y, w_1, w_2 \in \Sigma^*$, we have $x \leq y \Rightarrow w_1 x w_2 \leq w_1 y w_2$.

A word $x \in \Sigma^*$ is a *subword* of $y \in \Sigma^*$ if x can be obtained by deleting some letters on y. This defines a relation that is obviously transitive and we denote $\subseteq$ the *subword order* on Σ^*. Given two walks γ and γ', γ is a *subwalk* of γ', if γ can be obtained from γ' by deleting some edges. We can extend the $\subseteq$ order to labeled walks as follows: given two walks γ, γ' on the footprint G of $\mathcal{G}$, we note $\gamma \subseteq \gamma'$ if γ and γ' begin on the same node and end on the same node, and γ is a subwalk of γ'.

Given a date $t \in \mathcal{T}$ and a word x in Σ^*, we denote by $\mathcal{J}^*(t, x)$ the set $\{\mathcal{J} \in \mathcal{J}^*(\mathcal{G}) : start(\mathcal{J}) = t, \lambda(\mathcal{J}) = x\}$. $\mathcal{J}^*(x)$ denotes the set $\bigcup_{t \in \mathcal{T}} \mathcal{J}^*(t, x)$. Given a journey $\mathcal{J}$, $\bar{\mathcal{J}}$ is the corresponding labeled walk (in the footprint G). We denote by $\Gamma(x)$ the set $\{\bar{J} : \lambda(\mathcal{J}) = x\}$.

In the following, we consider only "complete" TVG (i.e. there exists a transition for each letter in each state.) so we have $\mathcal{J}^*(y)$ not empty for all word y; complete TVG can be obtained from any TVG (without changing the recognized language) by adding a sink node where any (missing) transition is sent. In this way, all words have at least one corresponding journey in the TVG.

Let x and y be two words in Σ^*. We define the quasi-order $\prec$, as follows: $x \prec y$ if

$$\forall \mathcal{J} \in \mathcal{J}^*(y), \exists \gamma \in \Gamma(x), \gamma \subseteq \bar{\mathcal{J}}.$$

The relation $\prec$ is obviously reflexive. We now establish the link between comparable words and their associated journeys and walks, and state some useful properties of relation $\prec$.

Lemma 4.2. *Let $x, y \in \Sigma^*$ be such that $x \prec y$. Then for any $\mathcal{J}_y \in \mathcal{J}^*(y)$, there exists $\mathcal{J}_x \in \mathcal{J}^*(x)$ such that $\bar{\mathcal{J}}_x \subseteq \bar{\mathcal{J}}_y$, $start(\mathcal{J}_x) = start(\mathcal{J}_y)$, $arrival(\mathcal{J}_x) = arrival(\mathcal{J}_y)$.*

Proof. By definition, there exists a labeled walk $\gamma \in \Gamma(x)$ such that $\gamma \subseteq \bar{\mathcal{J}}_y$. It is then possible to find a journey $\mathcal{J}_x \in \mathcal{J}^*(x)$ with $\bar{\mathcal{J}}_x = \gamma$, $start(\mathcal{J}_x) = start(\mathcal{J}_y)$ and $arrival(\mathcal{J}_x) = arrival(\mathcal{J}_y)$ by using for every edge of $\mathcal{J}_x$ the schedule of the same edge in $\mathcal{J}_y$. ☐

Proposition 4.3. *The relation $\prec$ is transitive.*

Proof. Suppose we have $x \prec y$ and $y \prec z$. Consider $\mathcal{J} \in \mathcal{J}^*(z)$. By Lemma 4.2, we get a journey $\mathcal{J}_y \in \mathcal{J}^*(y)$, such that $\bar{\mathcal{J}}_y \subseteq \bar{\mathcal{J}}$. By definition, there exists $\gamma \in \Gamma(x)$ such that $\gamma \subseteq \bar{\mathcal{J}}_y$. Therefore $\gamma \subseteq \bar{\mathcal{J}}$, and finally $x \prec z$. ☐

Let $L \subset \Sigma^*$. For any quasi-order $\leq$, we denote $\text{DOWN}_{\leq}(L) = \{x \mid \exists y \in L, x \leq y\}$.

The following is a corollary of Lemma 4.2:

Corollary 4.4. *Consider the language L of words induced by labels of journeys from u to v starting at time t. Then $\text{DOWN}_{\prec}(L) = L$.*

The following theorem is due to Harju and Ilie; this is a generalization of the well known theorem from Ehrenfeucht *et al* [26], which needs closure in the other (upper) direction.

Theorem 4.5 (Th. 6.3 [34])**.** *For any monotone well quasi order $\leq$ of Σ^*, for any $L \subset \Sigma^*$, the language $\text{DOWN}_{\leq}(L)$ is regular.*

The main proposition to be proved now is that $(\Sigma^*, \prec)$ is a well quasi-order (Proposition 4.12 below). We have first to prove the following.

Proposition 4.6. *The quasi-order $\prec$ is monotone.*

Proof. Let x, y be such that $x \prec y$. Let $z \in \Sigma^*$. Let $\mathcal{J} \in \mathcal{J}^*(yz)$. Then there exists $\mathcal{J}_y \in \mathcal{J}^*(y)$ and $\mathcal{J}_z \in \mathcal{J}^*(arrival(\mathcal{J}_y), z)$ such that the end node of $\mathcal{J}_y$ is the start node of $\mathcal{J}_z$. By Lemma 4.2, there exists $\mathcal{J}_x$ that ends in the same node as $\mathcal{J}_y$ and with the same *arrival* time. We can consider $\mathcal{J}'$ the concatenation of $\mathcal{J}_x$ and $\mathcal{J}_z$. By construction $\bar{\mathcal{J}}' \in \Gamma(xz)$, and $\bar{\mathcal{J}}' \subseteq \bar{\mathcal{J}}$. Therefore $xz \prec yz$. The property $zx \prec zy$ is proved similarly using the *start* property of Lemma 4.2. ☐

Proposition 4.7. *The quasi-order $\prec$ is well founded.*

Proof. Consider a descending chain $x_1 \succ x_2 \succ x_3 \succ \ldots$ such that for no i $x_i \prec x_{i+1}$. We show that this chain is finite. Suppose the contrary. By definition of $\prec$, we can find $\gamma_1, \gamma_2, \ldots$ such that for all i, $\gamma_i \in \bar{\mathcal{J}^*}(x_i)$, and such that $\gamma_{i+1} \subseteq \gamma_i$. This chain of walks is necessarily stationary and there exits i_0 such that $\gamma_{i_0} = \gamma_{i_0+1}$. Therefore, $x_{i_0} = x_{i_0+1}$, a contradiction. ☐

To prove that $\prec$ is a well quasi-order, we now have to prove that all antichains are finite. Let $(Q, \leq)$ be a quasi-order. For all $A, B \subset Q$, we denote $A \leq_{\mathcal{P}} B$ if there exists an injective mapping $\varphi : A \longrightarrow B$, such that for all $a \in A$, $a \leq \varphi(a)$. The relation $\leq_{\mathcal{P}}$ is transitive and defines a quasi-order on $\mathcal{P}(Q)$, the set of subsets of Q.

About the finiteness of antichains, we recall the following result

Lemma 4.8 ([35])**.** *Let $(Q, \leq)$ be a well quasi-order. Then $(\mathcal{P}(Q), \leq_{\mathcal{P}})$ is a well quasi-order.*

and the fundamental result of Higman:

Theorem 4.9 ([35])**.** *Let Σ be a finite alphabet. Then $(\Sigma^*, \subseteq)$ is a well quasi-order.*

This implies that our set of journey-induced walks is also a well quasi-order for $\subseteq$ as it can be seen as a special instance of Higman's Theorem about the subword order. We are now ready to prove that all antichains are finite. We prove this result by using a technique similar to the variation by [46] of the proof of [35].

Lemma 4.10. *Let X be an antichain of Σ^*. If the relation $\prec$ is a well quasi-order on $\textsc{Down}_{\prec}(X) \backslash X$ then X is finite or $\textsc{Down}_{\prec}(X) \backslash X = \emptyset$.*

Proof. We denote $Q = \textsc{Down}_{\prec}(X) \backslash X$, and suppose $Q \neq \emptyset$, and that Q is a well quasi-order for $\prec$. Therefore the product and the associated product order $(\Sigma \times Q, \prec_{\times})$ define also a well quasi-order. We consider $A = \{(a, x) \mid a \in \Sigma, x \in Q, ax \in X\}$. Because $\prec$ is monotone, for all $(a, x), (a', x') \in A$, $(a, x) \prec_{\times} (b, y) \Rightarrow ax \prec by$. Indeed, in this case $a = b$ and $x \prec y \Rightarrow ax \prec ay$. So A has to be an antichain of the well quasi-order $\Sigma \times Q$. Therefore A is finite. By construction, this implies that X is also finite. □

Theorem 4.11. *Let $L \subset \Sigma^*$ be an antichain for $\prec$. Then L is finite.*

Proof. Suppose we have an infinite antichain X_0. We apply recursively the previous lemma infinitely many times, that is there exists for all $i \in \mathbb{N}$, a set X_i that is also an infinite antichain of Σ^*, such that $X_{i+1} \subset \textsc{Down}_{\prec}(X_i) \backslash X_i$.

We remark that if we cannot apply the lemma infinitely many times that would mean that $X_k = \emptyset$ for some k. The length of words in X_0 would be bounded by k, hence in this case, finiteness of X_0 is also granted.

Finally, by definition of $\textsc{Down}_{\prec}$, for all $x \in X_{i+1}$, there exists $y \in X_i$ such that $x \prec y$, ie $x \subseteq y$. It is also possible to choose the elements x such that no pair is sharing a common y. So $X_{i+1} \subseteq_{\mathcal{P}} X_i$, and we have a infinite descending chain of $(\mathcal{P}(\Sigma^*), \subseteq_{\mathcal{P}})$. This would contradict Lemma 4.8. □

From Propositions 4.3, 4.6, 4.7 and Theorem 4.11 we have the last missing ingredient:

Proposition 4.12. $(\Sigma^*, \prec)$ *is a well quasi-order.*

Indeed, from Proposition 4.12, Proposition 4.6, Corollary 4.4, and Theorem 4.5, it immediately follows that $L_{wait}(\mathcal{G})$ is a regular language for any TVG $\mathcal{G}$; that is,

Theorem 4.13. *Let Φ be any class of functions containing the constant functions. Then $\mathcal{L}^{\Phi}_{wait} = \mathcal{R}$.*

5. Bounded Waiting Allowed

To better understand the expressive power of waiting, we now turn our attention to *bounded waiting*; that is when indirect journeys are considered feasible if and only if the pause between consecutive edges has a bounded duration $d > 0$. We restrict our study to the class of Turing-computable functions $\textsc{Turing}$. We examine the set $\mathcal{L}^{\textsc{Turing}}_{wait[d]}$ of all languages expressed by $\textsc{Turing}-$TVGs when waiting is allowed up to d time units, and prove the negative result that for any fixed $d \geq 0$, $\mathcal{L}^{\textsc{Turing}}_{wait[d]} = \mathcal{L}^{\textsc{Turing}}_{nowait}$. That is, the complexity of the environment is not affected by allowing waiting for a limited amount of time when the latency and presence are computable.

The basic idea is to reuse the same technique as in Section 3, but with a dilatation of time, i.e., given the bound d, the edge schedule is time-expanded by a factor greater than d (and thus no new choice of transitions is created compared to the no-waiting case).

Theorem 5.1. *For any duration d, $\mathcal{L}^{\textsc{Turing}}_{wait[d]} = \mathcal{L}^{\textsc{Turing}}_{nowait}$.*

Proof. Let L be an arbitrary $\textsc{Turing}-$recognizable language defined over the finite alphabet Σ. We denote by ψ its characteristic function. Let $d \in \mathbb{N}$ be the maximal waiting duration. We note $K = q^{1+\log_q(d)}$. We consider a TVG $\mathcal{G}_{2,d}$ structurally equivalent to $\mathcal{G}_2$ (see Figure 3 in Section 3), *i.e.*, $\mathcal{G}_{2,d} = (V, E, \mathcal{T}, \rho, \zeta)$ such that $V = \{v_0, v_1, v_2\}$, $E = \{\{(v_0, v_1, i), i \in \Sigma\} \cup \{\{(v_0, v_2, i), i \in \Sigma\}, \cup \{(v_1, v_1, i), i \in \Sigma\} \cup \{(v_1, v_2, i), i \in \Sigma\} \cup \{(v_2, v_1, i), i \in \Sigma\} \cup \{(v_2, v_2, i), i \in \Sigma\}\}$. The initial state is v_0, and the accepting state is v_1. If $\varepsilon \in L$ then v_0 is also accepting.

The presence and latency functions are now defined along the lines as those of $\mathcal{G}_2$, the only difference being that we are somehow stretching the time by a factor K.

For all $u \in \{v_0, v_1\}$, $i \in \Sigma$, and $t \geq 0$, we define

- $\rho((u, v_0, i), 0) = true$ iff $\psi_i(0) = c$
- $\zeta((u, v_1, i), 0) = K \times i$,
- $\rho((u, v_0, i), t) = true$ iff $\psi_i(\lfloor \frac{t}{K} \rfloor) = c$ and $\lfloor \frac{t}{K} \rfloor > 0$,
- $\zeta((u, v_0, i), t) = \psi_i(t) - \psi(t)$
- $\rho((u, v_1, i), t) = true$ iff $\psi_i(\lfloor \frac{t}{K} \rfloor) \neq c$
- $\zeta((u, v_1, i), t) = \psi_i(t) - \psi(t)$, $t \neq 0$.

First, this is indeed a TURING-TVG.

For any word w, we denote by n_w the corresponding integer (still using the q based enconding). By the same induction technique as in Section 3, we have that $L \subseteq L(\mathcal{G}_{2,d})$. Similarly, we have that any journey labeled by w ends at time exactly Kn_w, even if some $d-$waiting occurred. Finally, we remark that for all words $w, w' \in \Sigma^+$ such that $w \neq w'$, we have $|Kn_w - Kn_{w'}| \geq K > d$. Indeed, if $w \neq w'$ then they differ by at least one letter. The minimal time difference is when this is the first letter and these last letters are $i, i+1$ w.l.o.g. In this case, $|Kn_w - Kn_{w'}| \geq K$ by definition of ζ for $t = 0$. Therefore waiting for a duration of d does not enable more transitions in terms of labeling. $\square$

6. Concluding Remarks and Research Directions

We have studied the impact that waiting has on the expressivity of time-varying graphs, examining the difference in the type of languages expressed by indirect journeys (i.e., waiting is allowed) and direct journeys (i.e., waiting is unfeasible). We have shown that, if waiting is *not allowed*, then for any computable language L, there exists a time-varying graph $\mathcal{G}$, with computable functions for presence and latency, such that $L_{nowait}(\mathcal{G}) = L$. This result has to be compared with the fact that, as we have also proved, if waiting is *allowed*, then a TVG can express only regular languages, and this is even if the latency functions are arbitrarily complex (e.g., non-computable) functions of time.

In other words, if waiting is allowed, the difficulty of the language from arbitrary is always simplified to be regular. This expressivity gap can be rephrased as a computational gap: when the guards are (at least) Turing-computable, the power of the TVG automaton drops drastically from being (at least) as powerful as a Turing machine, to becoming that of a Finite-State Machine. Note that the result is also valid for continuous time models. In some sense, when considering the untimed behaviour (the trajectories), discrete systems are as expressive as continuous systems.

These results open interesting new research directions and pose intriguing questions, some listed in the following.

– *Language Classes.*

Several interesting problems are open on the relationship between TVG and language classes. In particular:

What restrictions on the journeys would characterize other classes of languages, e.g. only context-sensitive languages ?

For which computability class Φ the containment of the set $\mathcal{L}^{\Phi}_{nowait}$ in the set of $\Phi-$recognizable languages is strict ?

When waiting is allowed, what restrictions would identify specific subclasses of the class of regular languages ?

Can the equivalence of recognizable languages between 0-delay and d-delay TVG automaton be generalized to any $q-$stable computability class ?

– *Randomized extensions.*

In this paper we have considered time-varying graphs where all functions (presence, latency, waiting time) are *deterministic*.

An important research direction is to consider the impact on expressivity of non-deterministic settings. Interesting questions include, for example, the study of the expressivity of time-varying graphs where $\rho(e, t)$ is the probability that edge e exists at time t; or where the latency or the waiting time is a random function.

Indeed, the study of the expressivity of *random journeys* is an inviting open research direction.

– *Application in highly dynamic networks.*

Indirect and direct journeys in time-varying graphs correspond to the presence and absence, respectively, of unbounded buffering in highly dynamic networks. Obviously the availability of buffers (i.e., the ability to wait) increases the number of available journeys and thus offers more computational power to the designer of protocols for specific applications and tasks (broadcasting, routing, etc.).

The results established here, that $\mathcal{L}^{\Phi}_{wait}$ is regular while $\mathcal{L}^{\Phi}_{nowait}$ is a Φ language, provide a qualitative insight on the impact of buffering, rather than a quantitative measure. This leaves open the important research question of how to measure this computational impact. Indeed in a network modelled by $\mathcal{G}$, when waiting is allowed, the net gain in terms of of available journeys is precisely $\Delta(\mathcal{G}) = L_{wait}(\mathcal{G}) \setminus L_{nowait}(\mathcal{G})$. The quantitative study of these differences for classes of networks seems to be an important research direction. In this line of investigation, there are many interesting questions with possibly useful implications, e.g., to determine whether $\Delta(\mathcal{G}) = \emptyset$; i.e., whether or not $L_{wait}(\mathcal{G}) = L_{nowait}(\mathcal{G})$.

The insights our results provide on the nature of time-varying graphs do not seem to have an immediate practical impact on tasks and problems in highly dynamic networks. Thus the need for investigations on computability and complexity in time-varying graphs in presence of waiting is still pressing, both in general and for specific classes of problems (e.g., information diffusion, routing, etc.).

Acknowledgments. The authors would like to thank the anonymous referees for their helpful comments and questions.

This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada through the Discovery Grant program, by Prof. Flocchini's University Research Chair, by a Pacific Institute for Mathematical Science grant, and by the Scientific Grant in Aid by the Ministry of Education, Sports, Culture and Technology of Japan.

References

[1] E. Aaron, D. Krizanc, and E. Meyerson. DMVP: Foremost waypoint coverage of time-varying graphs. In *Proceedings of 40th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 29–41, 2014.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] M. Antony and A. Gupta. Finding a small set of high degree nodes in time-varying graphs. In *Proceedings 15th IEEE International Symposium on Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, 2014.

[4] L. Arantes, F. Greve, P. Sens, and V. Simon. Eventual leader election in evolving mobile networks. In *Proceedings of the 17th Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 23–37, 2013.

[5] C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world. In *Proceedings of the 35th Int. Colloquium on Automata, Languages and Programming (ICALP)*, pages 121–132, 2008.

[6] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *Proceedings 3rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 278–281. ACM, 1984.

[7] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proceedings 28th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 260–269, 2009.

[8] S. Bhadra and A. Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Proceedings* 2nd *Intl. Conference on Ad Hoc Networks and Wireless (ADHOC-NOW)*, pages 259–270, 2003.

[9] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings 19th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 73–84, 2012.

[10] A. Boutet, A.-M. Kermarrec, E. Le Merrer, and A. Van Kempen. On the impact of users availability in osns. In *Proceedings of 5th ACM Workshop on Social Network Systems (SNS)*, pages 4:1–4:6, 2012.

[11] P. Brandes and F. Meyer auf der Heide. Distributed computing in fault-prone dynamic networks. In *Proceedings of 4th International Workshop on Theoretical Aspects of Dynamic Distributed Systems (TADDS)*, pages 9–14, 2012.

[12] B. Brejova, S. Dobrev, R. Kralovic, and T. Vinar. Efficient routing in carrier-based mobile networks. *Theoretical Computer Science*, 509:113–121, 2013.

[13] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Intl. Journal of Foundations of Computer Science*, 14(2):267–285, April 2003.

[14] Q. Cai, J. Niu, and G. Qu. Identifying high dissemination capability nodes in opportunistic social networks. In *Proceedings IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4445–4450, 2013.

[15] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proceedings 16th Intl. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 126–140, 2009.

[16] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic computations in time-varying graphs: Broadcasting under unstructured mobility. In *Proceedings 5th IFIP Conference on Theoretical Computer Science (TCS)*, pages 111–124, 2010.

[17] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. *IEEE Transactions on Computers*, 63(2):397–410, 2014.

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

[19] A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1694–1712, 2010.

[20] A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1425–1432, 2011.

[21] A. Cornejo, C. Newport, S. Gollakota, J. Rao, and T.J. Giuli. Prioritized gossip in vehicular networks. *Ad Hoc Networks*, 11(1):397–409, 2013.

[22] E. Coulouma and E. Godard. A characterization of dynamic networks where consensus is solvable. In *Proceedings 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 24–35, 2013.

[23] O. Denysyuk and L. Rodrigues. Random walks on evolving graphs with recurring topologies. In *Proceedings 28th International Symposium on Distributed Computing (DISC)*, pages 333–345, 2014.

[24] C.A. Di Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. In *Proceedings 15th International Conference on Distributed Computing and Networking (ICDCN)*, pages 257–271, 2014.

[25] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736, 2013.

[26] A. Ehrenfeucht, D. Haussler, and G. Rozenberg. On regularity of context-free languages. *Theoretical Computer Science*, 27(3):311–332, 1983.

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

[28] P. Flocchini, M. Kellett, P.C. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012.

[29] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, January 2013.

[30] C. Gomez-Calzado, A. Lafuente, M. Larrea, and M. Raynal. Fault-tolerant leader election in mobile dynamic distributed systems. In *Proceedings 19th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 78–87, 2013.

[31] F. Greve, P. Sens, L. Arantes, and V. Simon. Eventually strong failure detector with unknown membership. *The Computer Journal*, 55(12):1507–1524, 2012.

[32] X.F. Guo and M.C. Chan. Change awareness in opportunistic networks. In *Proceedings 10th IEEE Int. Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, pages 365–373, 2013.

[33] F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–88, 1997.

[34] T. Harju and L. Ilie. On quasi orders of words and the confluence property. *Theoretical Computer Science*, 200(1-2):205–224, 1998.

[35] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2:326–336, 1952.

[36] D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *Proceedings 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 250–262, 2014.

[37] D. Ilcinkas and A. Wade. On the power of waiting when exploring public transportation systems. *Proceedings 15th Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 451–464, 2011.

[38] E.P.C. Jones, L. Li, J.K. Schmidtke, and P.A.S. Ward. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 6(8):943–959, 2007.

[39] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proceedings 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, 2002.

[40] M. Konschake, H.H.K. Lentz, F.J. Conraths, P. Hovel, and T. Selhorst. On the robustness of in-and out-components in a temporal network. *PloS One*, 8(2):e55223, 2013.

[41] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network. In *Proceedings of the 14th Int. Conference on Knowledge Discovery and Data Mining (KDD)*, pages 435–443, 2008.

[42] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings 42nd ACM Symposium on Theory of Computing (STOC)*, pages 513–522. ACM, 2010.

[43] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *Proceedings 30th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–10. ACM, 2011.

[44] C. Liu and J. Wu. Scalable routing in cyclic mobile networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(9):1325–1338, 2009.

[45] O. Michail, I. Chatzigiannakis, and P.G.. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74(1):20162026, 2014.

[46] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 59(04):833–835, 1963.

[47] F.J. Ros and P.M. Ruiz. Minimum broadcasting structure for optimal data dissemination in vehicular networks. *IEEE Transactions on Vehicular Technology*, 62(8):3964–3973, 2013.

[48] J. Tang, S. Scellato, M. Musolesi, C. Mascolo, and V. Latora. Small-world behavior in time-varying graphs. *Physical Review E*, 81(5):055101, 2010.

[49] J. Whitbeck, M. Dias de Amorim, V. Conan, and J.-L. Guillaume. Temporal reachability graphs. In *Proceedings 8th International Conference on Mobile Computing and Networking (MOBICOM)*, pages 377–388, 2012.

[50] Z. Yang, S. Yat-sen, W. Wu, Y. Chen, and J. Zhang. Efficient information dissemination in dynamic networks. In *Proceedings of 42nd International Conference on Parallel Processing (ICPP)*, pages 603–610, 2013.

[51] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.