Theoretical Computer Science 590 (2015) 27–37

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

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

ARTICLE INFO

Article history:
Received 21 December 2013
Received in revised form 17 November 2014
Accepted 3 April 2015
Available online 9 April 2015

Keywords:
Time-varying graphs
Dynamic networks
Buffering
Expressivity of TVGs
Automata
Language

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 modeled 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.

* Corresponding author.
E-mail address: emmanuel.godard@lif.univ-mrs.fr (E. Godard).

http://dx.doi.org/10.1016/j.tcs.2015.04.004
0304-3975/

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 modeled 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] $< e_1, e_2, \ldots, e_k >$ with a sequence of time instants $< t_1, t_2, \ldots, t_k >$ 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 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.

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