Discrete Mathematics 339 (2016) 2168–2177

Contents lists available at ScienceDirect

Discrete Mathematics

journal homepage: www.elsevier.com/locate/disc

Linearity is strictly more powerful than contiguity for encoding graphs[☆,☆☆]

Christophe Crespelle[a,b], Tien-Nam Le[c], Kevin Perrot[d,e], Thi Ha Duong Phan[b]

[a] *Université Claude Bernard Lyon 1 and CNRS, DANTE/INRIA, LIP UMR CNRS 5668, ENS de Lyon, Université de Lyon, France*
[b] *Institute of Mathematics, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet, Hanoi, Viet Nam*
[c] *ENS de Lyon, Université de Lyon, France*
[d] *Universidad de Chile, DIM, CMM UMR CNRS 2807, Santiago, Chile*
[e] *Aix-Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille, France*

ARTICLE INFO

Article history:
Received 24 July 2015
Accepted 6 March 2016
Available online 26 April 2016

Keywords:
Graph encoding
Linearity
Contiguity
Cographs

ABSTRACT

Linearity and contiguity are two parameters devoted to graph encoding. Linearity is a generalization of contiguity in the sense that every encoding achieving contiguity k induces an encoding achieving linearity k, both encoding having size $\Theta(k.n)$, where n is the number of vertices of G. In this paper, we prove that linearity is a strictly more powerful encoding than contiguity, *i.e.* there exists some graph family such that the linearity is asymptotically negligible in front of the contiguity. We prove this by answering an open question asking for the worst case linearity of a cograph on n vertices: we provide an $O(\log n/\log\log n)$ upper bound which matches the previously known lower bound.

1. Introduction

One of the most widely used operation in graph algorithms is the *neighbourhood query*: given a vertex x of a graph G, one wants to obtain the list of neighbours of x in G. The classical data structure that allows to do so is the adjacency lists. It stores a graph G in $O(n+m)$ space, where n is the number of vertices of G and m its number of edges, and answers a neighbourhood query on any vertex x in $O(d)$ time, where d is the degree of vertex x. This time complexity is optimal, as long as one wants to produce the list of neighbours of x. On the other hand, in the last decades, huge amounts of data organized in the form of graphs or networks have appeared in many contexts such as genomic, biology, physics, linguistics, computer science, transportation and industry. In the same time, the need, for industrials and academics, to algorithmically treat this data in order to extract relevant information has grown in the same proportions. For these applications dealing with very large graphs, a space complexity of $O(n+m)$ is often very limiting. Therefore, as pointed out by [15], finding compact representations of a graph providing optimal time neighbourhood queries is a crucial issue in practice. Such representations allow to store the graph entirely in memory while preserving the complexity of algorithms using neighbourhood queries. The conjunction of these two advantages has great impact on the running time of algorithms managing large amount of data.

☆ This work was partially funded by a grant from Région Rhône-Alpes, the delegation program of CNRS, the Vietnam Institute for Advanced Study in Mathematics (VIASM), the Vietnam National Foundation for Science and Technology Development (NAFOSTED), Fondecyt Postdoctoral grant 3140527 and Núcleo Milenio Información y Coordinación en Redes (ACGO).

☆☆ This is a complete version of the extended abstract appeared in WADS 2015 (Crespelle et al., 2015) [9].

E-mail addresses: christophe.crespelle@inria.fr (C. Crespelle), tien-nam.le@ens-lyon.fr (T.-N. Le), kevin.perrot@lif.univ-mrs.fr (K. Perrot), phanhaduong@math.ac.vn (T.H.D. Phan).

http://dx.doi.org/10.1016/j.disc.2016.03.006
0012-365X/© 2016 Elsevier B.V. All rights reserved.

One possible way to store a graph G in a very compact way and preserve the complexity of neighbourhood queries is to find an order σ on the vertices of G such that the neighbourhood of each vertex x of G is an interval in σ. In this way, one can store the order σ on the vertices of G and assign two pointers to each vertex: one towards its first neighbour in σ and one towards its last neighbour in σ. Therefore, one can answer adjacency queries on vertex x simply by listing the vertices appearing in σ between its first and last pointer. It must be clear that such an order on the vertices of G does not exist for all graphs G. Nevertheless, this idea turns out to be quite efficient in practice and some compression techniques are precisely based on it [1,3,4,2,13]: they try to find orders on the vertices that group the neighbourhoods together, as much as possible.

Then, a natural way to relax the constraints of the problem so that it admits a solution for a larger class of graphs is to allow the neighbourhood of each vertex to be split in at most k intervals in order σ. The minimum value of k which makes possible to encode the graph G in this way is a parameter called *contiguity* [11] and denoted by $cont(G)$. Another natural way of generalization is to use at most k orders $\sigma_1, \ldots, \sigma_k$ on the vertices of G such that the neighbourhood of each vertex is the union of exactly one interval taken in each of the k orders. This defines a parameter called the *linearity* of G [6], denoted $lin(G)$. The additional flexibility offered by linearity (using k orders instead of just 1) results in a greater power of encoding, in the sense that if a graph G admits an encoding by contiguity k, using one linear order σ and at most k intervals for each vertex, it is straightforward to obtain an encoding of G by linearity k: take k copies of σ and assign to each vertex one of its k intervals in each of the k copies of σ.

As one can expect, this greater power of encoding requires an extra cost: the size of an encoding by linearity k, which uses k orders, is greater than the size of an encoding by contiguity k, which uses only 1 order. Nevertheless, very interestingly, the sizes of these two encodings are equivalent up to a multiplicative constant. Indeed, storing an encoding by contiguity k requires to store a linear ordering of the n vertices of G, *i.e.* a list of n integers, and the bounds of each of the k intervals for each vertex, *i.e.* $2kn$ integers, the total size of the encoding being $(2k + 1)n$ integers. On the other hand, the linearity encoding also requires to store $2kn$ integers for the bounds of the k intervals of each vertex, but it needs k linear orderings of the vertices instead of just one, that is kn integers. Thus, the total size of an encoding by linearity k is $3kn$ integers instead of $(2k + 1)n$ for contiguity k and therefore the two encodings have equivalent sizes.

Then the question naturally arises to know whether there are some graphs for which the linearity is significantly less than the contiguity. More formally, does there exist some graph family for which the linearity is asymptotically negligible in front of the contiguity? Or are these two parameters equivalent up to a multiplicative constant? This is the question we address here. Our results show that linearity is strictly more powerful than contiguity.

Related work. Only little is known about contiguity and linearity of graphs. In the context of $0-1$ matrices, [11,16] studied closed contiguity and showed that deciding whether an arbitrary graph has closed contiguity at most k is NP-complete for any fixed $k \geq 2$. For arbitrary graphs again, [10] (Corollary 3.4) gave an upper bound on the value of closed contiguity which is $n/4 + O(\sqrt{n \log n})$. Regarding graphs with bounded contiguity or linearity, only the class of graphs having contiguity 1, or equivalently linearity 1, has been characterized, as being the class of proper (or unit) interval graphs [14]. For interval graphs and permutation graphs, [6] showed that both contiguity and linearity can be up to $\Omega(\log n/ \log\log n)$. For cographs, a subclass of permutation graphs, [8] showed that the contiguity can even be up to $\Omega(\log n)$ and is always $O(\log n)$, implying that both bounds are tight. The $O(\log n)$ upper bound consequently applies for the linearity (of cographs) as well, but [8] only provides an $\Omega(\log n/ \log\log n)$ lower bound. Finally, let us mention for the sake of completeness that [7] gave an algorithm that computes a constant ratio approximation of the contiguity of a cograph, as well as a corresponding encoding, in linear time.

Our results. Our main result (Theorem 2) is to exhibit a family of graphs G_h, $h \geq 1$, such that the linearity of G_h is asymptotically negligible in front of the contiguity of G_h. In order to do so, we prove (Theorem 1) that the linearity of a cograph G on n vertices is always $O(\log n/ \log\log n)$. It turns out that this bound is tight, as it matches the previously known lower bound on the worst-case linearity of a cograph [8].

Outline of the paper. Section 2 gives necessary background on the notions used throughout the article. Section 3 proves the key technical statement of our work, showing that the linearity of a cograph is dominated by the maximal height of a certain type of tree, called *double factorial tree*, included in its cotree. From there, Section 4 derives our main results: the tight upper bound on the linearity of cographs and the construction of a subfamily of cographs for which the linearity is negligible in front of the contiguity.

2. Preliminaries

All graphs considered here are finite, undirected, simple and loopless. In the following, G is a graph, V (or $V(G)$) is its vertex set and E (or $E(G)$) is its edge set. We use the notation $G = (V, E)$ and n stands for the cardinality $|V|$ of $V(G)$. An edge between vertices x and y will be arbitrarily denoted by xy or yx. The (open) neighbourhood of x is denoted by $N(x)$ (or $N_G(x)$) and its closed neighbourhood by $N[x] = N(x) \cup \{x\}$. The subgraph of G induced by the set of vertices $X \subseteq V$ is denoted by $G[X] = (X, \{xy \in E \mid x, y \in X\})$.

For a rooted tree T and a node $u \in T$, the depth of u in T is the number of edges in the path from the root of T to u (the root has depth 0). The *height* of T, denoted by $h(T)$, is the greatest depth of its leaves. We employ the usual terminology for *children*, *father*, *ancestors* and *descendants* of a node u in T (the two later notions including u itself), and denote by $\mathcal{C}(u)$ the