Linearity is Strictly More Powerful than Contiguity for Encoding Graphs*†‡

Christophe Crespelle§ Tien-Nam Le¶ Kevin Perrot‖
Thi Ha Duong Phan**

Abstract

Linearity and contiguity are two parameters devoted to graph encoding. Linearity is a generalisation 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,

*This work was partially funded by a grant from Région Rhône-Alpes and by the delegation program of CNRS.

†This work was partially funded by the Vietnam Institute for Advanced Study in Mathematics (VIASM) and by the Vietnam National Fondation for Science and Technology Developement (NAFOSTED).

‡This work was partially funded by Fondecyt Postdoctoral grant 3140527 and Núcleo Milenio Información y Coordinación en Redes (ACGO).

§Université Claude Bernard Lyon 1 and CNRS, DANTE/INRIA, LIP UMR CNRS 5668, ENS de Lyon, Université de Lyon; and Institute of Mathematics, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet, Hanoi, Vietnam – christophe.crespelle@inria.fr

¶ENS de Lyon, Université de Lyon – tien-nam.le@ens-lyon.fr

‖Universidad de Chile, DIM, CMM UMR CNRS 2807, Santiago, Chile; and Aix-Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille, France – kevin.perrot@lif.univ-mrs.fr

**Institute of Mathematics, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet, Hanoi, Vietnam – phanhaduong@math.ac.vn

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 [13], 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.

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 toward its first neighbour in σ and one toward 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, 2, 3, 4, 11]: they try to find orders of 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* [9] 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, [9, 14] 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, [8] (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 [12]. 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, [7] showed that the contiguity can even been 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 [7] only provides an $\Omega(\log n / \log\log n)$ lower bound.

Our results. Our main result (Corollary 1) 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 [7].

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 set of children of u. The subtree of T rooted at u, denoted by T_u, is the tree induced by node u and all its descendants in T. A *monotonic path* C of a rooted tree T is a path such that there exists some node $u \in C$ such that all nodes of C are ancestors of u. The unique node of C which has no parent in C is called the root of the monotonic path.

In the following, the notion of *minors* of rooted trees is central. This is a special case of minors of graphs (see e.g. [10]), for which we give a simplified definition in the context of rooted trees. The *contraction of edge* uv in a rooted tree T, where u is the parent of v, consists in removing v from T and assigning its children (if any) to node u.

Definition 1 (Minor) *A rooted tree T' is a* minor *of a rooted tree T if it can be obtained from T by a sequence of edge contractions.*

There are actually two notions of linearity depending on whether one uses the open neighbourhood $N(x)$ or closed neighbourhood $N[x]$.

Definition 2 (p-line-model) *A* closed p-line-model *(resp. open p-line-model) of a graph $G = (V, E)$ is a tuple $(\sigma_1, \ldots, \sigma_p)$ of linear orders on V such that $\forall v \in V, \exists (I_1, \ldots, I_p)$ such that $\forall i \in [\![1, p]\!]$, I_i is an interval of σ_i and $N[x] = \bigcup_{1 \leq i \leq p} I_i$ (resp. $N(x) = \bigcup_{1 \leq i \leq p} I_i$). The* closed linearity *(resp.* open linearity*) of G, denoted by $cl(G)$ (resp. $ol(G)$), is the minimum integer p such that there exists a closed p-line-model (resp. open p-line-model) of G.*

Remark 1 *In the definition of a p-line-model, the set of vertices of the intervals I_i assigned to a vertex x are not necessarily disjoint. They are only required to cover the neighbourhood of x while being included in it.*

In all the paper, we abusively extend the notion of linearity to cotrees, referring to the linearity of their associated cograph. Moreover, we consider only closed linearity but, from the inequalities below, the bounds we obtain (which hold up to multiplicative constants) also hold for the open linearity. Then, for the sake of clarity, as we will not use the open notion, in the following, we denote $lin(G)$ instead of $cl(G)$.

Lemma 1 *For an arbitrary graph G, we have the following inequalities: $cl(G) - 1 \leq ol(G) \leq 2cl(G)$.*

There are several characterizations of the class of *cographs*. They are often defined as the graphs that do not admit the P_4 (path on 4 vertices) as induced subgraph. Equivalently, they are the graphs obtained from a single vertex under the closure of the parallel composition and the series composition. The parallel composition of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the disjoint union of G_1 and G_2, i.e., the graph $G_{par} = \big(V_1 \cup V_2, E_1 \cup E_2\big)$. The series composition of two graphs G_1 and G_2 is the disjoint union of G_1 and G_2 plus all possible edges from a vertex of G_1 to one of G_2, i.e., the graph $G_{ser}\big(V_1 \cup V_2, E_1 \cup E_2 \cup \{xy \mid x \in V_1, y \in V_2\}\big)$. These operations can naturally be extended to a finite number of graphs.

This gives a very nice representation of a cograph G by a tree whose leaves are the vertices of the graph and whose internal nodes (non-leaf nodes) are labelled P, for parallel, or S, for series, corresponding to the operations used in the construction of G. It is always possible to find such a labelled tree T representing G such that every internal node has at least two children, no two parallel nodes are adjacent in T and no two series nodes are adjacent. This tree T is unique [5] and is called the *cotree* of G. Note that the subtree T_u rooted at some node u of cotree T also defines a cograph, denoted G_u, and then $V(G_u)$ is the set of leaves of T_u. The adjacencies between vertices of a cograph can easily be read on its cotree, in the following way.

Remark 2 *Two vertices x and y of a cograph G having cotree T are adjacent iff the least common ancestor u of leaves x and y in T is a series node. Otherwise, if u is a parallel node, x and y are not adjacent.*

For a graph encoding scheme Enc and a graph G, we denote $|Enc(G)|$ the minimum size of an encoding of G based on Enc. We now give a formal definition for an encoding scheme to be strictly more powerful than another one.

Definition 3 (Strictly more powerful encoding) *Let Enc_1 and Enc_2 be two graph encoding schemes. We say that Enc_2 is* at least as powerful as *Enc_1 iff there exists $\alpha > 0$ such that for all graphs G, $|Enc_2(G)| \leq \alpha |Enc_1(G)|$. Moreover, we say that Enc_2 is* strictly more powerful than *Enc_1 iff Enc_2 is at least as powerful as Enc_1 and the converse is not true.*

Note that, Enc_1 is not at least as powerful as Enc_2 iff there exists a series of graphs G_h, $h \geq 1$, such that $|Enc_1(G_h)|/|Enc_2(G_h)|$ tends to infinity when h tends to infinity. In the introduction, we showed that the encoding schemes $LinEnc$ and $ContEnc$ based on linearity and contiguity respectively are such that, for any graph G on n vertices, we have $2\,n\,cont(G) \leq |ContEnc(G)| \leq 3\,n\,cont(G)$ and $|LinEnc(G)| = 3\,n\,lin(G)$. Since $lin(G) \leq cont(G)$, this gives $|LinEnc(G)| \leq \frac{3}{2}|ContEnc(G)|$. In addition, the previous inequalities also imply that $\frac{2}{3}cont(G)/lin(G) \leq |ContEnc(G)|/|LinEnc(G)| \leq cont(G)/lin(G)$. Altogether, we obtain the following remark.

Remark 3 *Linearity is an encoding at least as powerful as contiguity according to Definition 3. Moreover, it is strictly more powerful iff there exists a series of graphs G_h, $h \geq 1$, such that $|cont(G_h)|/|lin(G_h)|$ tends to infinity when h tends to infinity.*

3 Linearity of a cograph and factorial rank of its cotree

In this section, we show that the linearity of a cograph is upper bounded by the size of some maximal structure contained in its cotree, more precisely by the height of a maximal double factorial tree (defined below), which we call the factorial rank of a cotree. This result is interesting by itself as it provides a structural explanation of the difficulty of encoding a cograph by linearity. For our concern, the interesting point is that the number of leaves of a double factorial tree of height h is $\Omega(h!)$. Combined with this fact, the result presented in this section (Lemma 2) will allow us to derive in next section the desired $O(\log n / \log\log n)$ upper bound on the linearity of cographs. We start by some necessary definitions.

Definition 4 (Double factorial tree) *The* double factorial tree *F^h of height h is defined inductively as the tree whose root has $2h+1$ children u, whose subtrees F_u are precisely F^{h-1}, F^0 being the unique tree of height 0 (i.e., made of a single leaf node).*

Definition 5 (Factorial rank) *The* factorial rank *of a rooted tree T denoted $factrank(T)$, is the maximum height of a double factorial tree being a minor of T, that is:*
$factrank(T) = \max\{h(T') \mid T'$ is a double factorial tree and a minor of $T\}$.

We extend the notion of factorial rank to a node, referring to the factorial rank of its subtree. The case where the children of node u all have factorial rank strictly less than the one of u will play a key role.

Definition 6 (Minimally of factorial rank k) *Let u be a node of a tree T. If u has factorial rank k and if all the children of u have factorial rank at most $k-1$, we say that u is* minimally of factorial rank k.

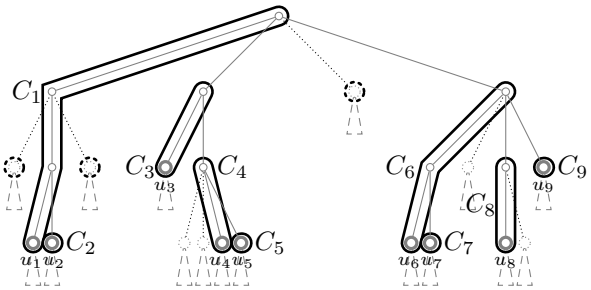

Figure 1: Example of partition into monotonic paths in the case where u is of factorial rank k. The three dot circled nodes of $U_{\leq k-1}$ form the set $U^1_{\leq k-1}$.

We are now ready to state the result of this section, which claims that the linearity of a cograph is linearly bounded by the factorial rank of its cotree.

Lemma 2 *Let T be a cotree and let $u \in T$ of factorial rank $k \geq 0$. Then, $lin(G_u) \leq 2k+1$. Moreover, if $k \geq 1$ and u is minimally of factorial rank k, then $lin(G_u) \leq 2k$.*

Sketch of proof. We prove the result by induction. We consider an integer $k \geq 1$ such that: all nodes of factorial rank $j \leq k-1$ have linearity at most $2j+1$; and all nodes which are minimally of factorial rank k (i.e., whose children have factorial rank at most $k-1$) have linearity at most $2k$. Then, we show that any node u of factorial rank k (not necessarily minimally) can be encoded using one more order (i.e. $2k+1$) and that adding again one more order (i.e. using $2k+2$ orders), we can also encode any node v which is minimally of factorial rank $k+1$.

Node u of factorial rank k. In order to describe a $2k+1$-line-model of G_u we need to distinguish different parts of T_u. Let U_k be the subset of nodes of T_u having factorial rank k and consider the set $U_{kmin} = \{u_1, u_2, \ldots, u_l\} \subsetneq U_k$ of its minimal elements for the ancestor relationship (i.e. the lowest in the cotree). Note that $|U_{kmin}| = l \leq 2k$, as otherwise u would be of factorial rank $k+1$ (since it would have $2k+1$ independent descendants of rank $2k$). By definition, all the children of the nodes of U_{kmin} have factorial rank at most $k-1$, and then the nodes of U_{kmin} are minimally of rank k. By induction hypothesis, it follows that for all $i \in [\![1, l]\!]$, u_i admits a $2k$-line-model for which we denote $\sigma_j(u_i)$, with $1 \leq j \leq 2k$, its $2k$ orders. We denote T'_u the subtree of T_u induced by the set of nodes U_k (by definition, $U_{kmin} \subseteq T'_u$). We also denote $U_{\leq k-1}$ the set of nodes of $T_u \setminus T'_u$ whose parent is in $T'_u \setminus U_{kmin}$. Nodes of $U_{\leq k-1}$ have, by definition, rank at most $k-1$ and it follows from the induction hypothesis that they admit a $(2k-1)$-line-model. Then, for a node $w \in U_{\leq k-1}$, we again denote $\sigma_j(w)$, with $1 \leq j \leq 2k-1$, the $2k-1$ orders of such a model. In addition , we use a partition $\mathcal{P}$ of the nodes of T'_u into l monotonic paths C_i such that for all $i \in [\![1, l]\!]$, $u_i \in C_i$ (see Figure 1). Partition $\mathcal{P}$ naturally induces a generalised partition (some parts may be empty) of $U_{\leq k-1}$ whose parts are the subset of nodes $U^i_{\leq k-1}$ of $U_{\leq k-1}$ whose parent belongs to $C_i \setminus \{u_i\}$.

We can now describe the $2k+1$ orders $(\sigma_j)_{1 \leq j \leq 2k+1}$ of the model we build for G_u. Importantly, note that $V(G_w)$, $w \in U_{kmin} \cup U_{\leq k-1}$, is a partition of $V(G_u)$. In our construction, $V(G_w)$ will always be an interval of σ_j for all $w \in U_{kmin} \cup U_{\leq k-1}$ and all $j \in [\![1, 2k+1]\!]$. Then, the description of σ_j is in two steps: we first give the order, denoted

π_j, in which the intervals of nodes $w \in U_{kmin} \cup U_{\leq k-1}$ appear in σ_j and then, for each w, we give the order, denoted σ_j^w, in which the vertices of G_w appear in this interval. The description of orders π_j will be done by choosing a local order on the children of each node of $U_k \setminus U_{kmin}$. Then π_j is defined as the unique order on $U_{kmin} \cup U_{\leq k-1}$ respecting all the chosen local orders, i.e. such that for any $v, v' \in U_{kmin} \cup U_{\leq k-1}$, if v and v' has the same parent z and if v comes before v' in the order chosen on children of z, then all descendants of v comes before all descendants of v' in π_j.

To fully describe the $2k+1$-line-model of u, we must also assign to each vertex x one interval of its neighbours in each of the orders of the model, in such a way that these intervals entirely cover the neighbourhood of x. In order to help our analysis, we distinguish between the *external neighbourhood* of node x, which is $N[x] \setminus V(G_w)$, where w is the unique node of $U_{kmin} \cup U_{\leq k-1}$ being an ancestor of leaf x in T_u, and its *internal neighbourhood* $N[x] \cap V(G_w)$. Our construction mainly focusses on the $2k$ first orders of the model, which we use to encode the majority of adjacencies of G_u, order σ_{2k+1} being used to encode the remaining ones.

For $j \in [\![1, 2k]\!]$, the purpose of order σ_j is to satisfy the external neighbourhoods of vertices of G_w for $w \in \{u_j\} \cup U^j_{\leq k-1}$. It entirely succeeds to do so for u_j and encodes only half of the external neighbourhoods of $V(G_w)$ for nodes $w \in U^j_{\leq k-1}$, the other half being encoded in σ_{2k+1}. Then, for each $w \in \{u_j\} \cup U^j_{\leq k-1}$, the internal neighbourhoods of vertices of G_w are encoded in the remaining $2k-1$ orders of $(\sigma_j)_{1 \leq j \leq 2k}$. It is enough for $w \in U^j_{\leq k-1}$, since they admit a $2k-1$-line-model by recursion hypothesis, but one order is missing for u_j which is minimally of linearity k and is then only guaranteed to admit a $2k$-line model by recursion hypothesis. Again, the missing order will be found in σ_{2k+1}.

External neighbourhoods and choice of π_j's. Let us now show how to choose the order π_j used for defining σ_j such that, as claimed above, most of the external adjacencies of vertices of G_w, for $w \in \{u_j\} \cup U^j_{\leq k-1}$, will be satisfied in σ_j. We choose π_j the order induced by the following local orders on the children of nodes $u' \in U_k \setminus U_{kmin}$: if u' is a series node (resp. parallel node) and a strict ancestor of u_i, then the child of u' which is an ancestor of u_j is placed first (resp. last) in the order on the children of u' (the order on the other children of u' does not matter), in all other cases, the order on the children of u' does not matter. This way, the external neighbourhood of vertices of G_{u_j} is an interval at the end of σ_j (the interval following G_{u_j}) and this is the interval assigned to vertices of G_{u_j} in σ_j. For nodes $w \in U^j_{\leq k-1}$ whose parent (which is a strict ancestor of u_j by definition) is a parallel node, the situation is the same. But for nodes $w \in U^j_{\leq k-1}$ whose parent w' is series, their external neighbourhood is split into two intervals of σ_j: one following $V(G_w)$, which is the one we assign to vertices of G_w in σ_j, and one preceding $V(G_w)$, denoted $I_{<w}$, which is constituted by the leaves of T_u descending from the children of w' that precede w in the order chosen for π_j.

This is where we need order σ_{2k+1} and the partition of T'_u into paths C_i introduced earlier. To define order π_{2k+1}, for any node $u' \in U_k \setminus U_{kmin}$, we use the same order on the children of u' as the one used for π_i, with $i \in [\![1, l]\!]$ such that $u' \in C_i$. This ensures that for any node $w \in U_{\leq k-1}$ whose parent w' is a series node of C_i, the interval $I_{<w}$ of external neighbours which was not covered in order σ_i (note that since $w' \in C_i$ then $w \in U^i_{\leq k-1}$) will also be an interval of σ_{2k+1}. This is precisely the interval we assign to vertices of G_w in σ_{2k+1}, which is possible as their internal neighbourhood will be entirely satisfied in the

$2k$ first orders, as described below.

Internal neighbourhoods and choice of σ_j^w's. The orders σ_j^w used for the vertices of G_w, with $w \in U_{kmin} \cup U_{\leq k-1}$, in order σ_j, with $j \in [\![1, 2k]\!]$ are chosen as follows. For a node $w \in U_{\leq k-1}$ whose parent belong to path C_i of the partition, if $j < i$ (resp. if $j > i$) then we use the order $\sigma_j(w)$ (resp. $\sigma_{j-1}(w)$), and the interval of σ_j associated to the vertices of G_w is the same as the one associated to them in $\sigma_j(w)$ (resp. $\sigma_{j-1}(w)$). Otherwise, if $j = i$ the order chosen for vertices of G_w does not matter as σ_j is used only for satisfying their external neighbourhood, see above. Proceeding this way, the internal neighbourhoods of vertices of G_w are entirely satisfied in orders $(\sigma_j)_{j\in[\![1,2k]\!]}$. For a node $u_i \in U_{kmin}$, if $j \neq i$, the order chosen on the vertices of G_{u_i} is $\sigma_j(u_i)$ and the interval associated to vertices of G_{u_i} in σ_j is the same as the one associated to them in $\sigma_j(u_i)$. Otherwise, if $j = i$, the order chosen for vertices of G_{u_i} does not matter again as σ_j is used only for satisfying their external neighbourhood. Then, only $2k - 1$ orders among the $2k$ first ones are used to encode the internal neighbourhoods of G_{u_i}, while the recursion hypothesis only guarantees that $lin(G_{u_i}) \leq 2k$. For this reason, we chose the order on the vertices of G_{u_i} in σ_{2k+1} as being $\sigma_i(u_i)$, the one which was not used until now, and the interval associated to vertices of G_{u_i} in σ_{2k+1} is the same as the one associated to them in $\sigma_i(u_i)$. This is possible as the external neighbourhood of vertices of G_{u_i} has already been entirely satisfied before, in order σ_i. Then, all adjacencies are satisfied and $lin(G_u) \leq 2k + 1$.

Node v minimally of factorial rank $k + 1$. The only interesting case is when v is a series node (the result is straightforward when v is parallel), then we denote $v_1, v_2, \ldots, v_l$, with $l \in \mathbb{N}$, the children of v, which have factorial rank at most k by definition. From what precedes, each of them v_i admit a $(2k + 1)$-line-model denoted $(\sigma_j(v_i))_{j\in[\![1,2k+1]\!]}$. A remarkable property of this $(2k + 1)$-line-model, which we have constructed above, is that for any vertex x, there exists an index j, later denoted $ind(x)$, such that the interval associated to x in $\sigma_j(v_i)$ contains the last vertex of $\sigma_j(v_i)$. Based on this, the model $(\sigma_j)_{1\leq j\leq 2k+2}$ we build for G_v is as follows. For $j \in [\![1, 2k + 1]\!]$, order σ_j is the concatenation of orders $\sigma_j(v_i)$ in the order from $i = 1$ to $i = l$. For any vertex x of G_{v_i}, if $j \neq ind(x)$, the interval associated to x in σ_j is the same as the one associated to x in $\sigma_j(v_i)$; and if $j = ind(x)$, as the interval associated to x in $\sigma_{ind(x)}(v_i)$ contains the last vertex of $\sigma_{ind(x)}(v_i)$, in the order $\sigma_{ind(x)}$ of the model of G_v, we extend this interval on the right by including the vertices of $G_{v_{i'}}$ for all $i' > i$. As v is a series node, all these vertices are indeed adjacent to x, as well as all the vertices of $G_{v_{i'}}$ for all $i' < i$, which are the only adjacencies of x that are not covered in the orders $(\sigma_j)_{1\leq j\leq 2k+1}$. We use order σ_{2k+2} to cover these adjacencies in the following way. For each node v_i, we choose an arbitrary order on the vertices of G_{v_i} and concatenate them in the order from $i = 1$ to $i = l$. Then, to any vertex x of G_{v_i}, we associate the interval made by all the vertices of $G_{v_{i'}}$ for all $i' < i$. This completes the $2k + 2$-model of v and the proof of the lemma. □

4 Main results

The first result we derive from Lemma 2 is a tight upper bound on the worst-case linearity of cographs on n vertices. Until now, the best known upper bound [7] was $O(\log n)$, and

[7] also exhibits some cograph families having a linearity up to $\Omega(\log n/\log\log n)$. Here, we show a new upper bound of $O(\log n/\log\log n)$ that matches the lower bound of [7]. This is a direct consequence of Lemma 2 and of the fact that a double factorial tree of height h has $\Omega(h!)$ vertices.

Theorem 1 *For any cograph G on n vertices, we have $lin(G) = O(\log n/\log\log n)$, and this upper bound is tight.*

Proof. Let T denote the cotree of G and $k = factrank(T)$. From Lemma 2, the linearity of G is in $O(k)$. Let us now show that $k = O(\log n/\log\log n)$, which will conclude this proof. According to the definition of factorial rank, G has at least as many vertices as the double factorial tree of height k, which has $\prod_{i=0}^{k}(2i+1)$ vertices. It follows from Stirling's approximation of factorial that

$$n \geq \prod_{i=0}^{k}(2i+1) = \frac{(2(k+1))!}{2^{k+1}(k+1)!} \geq \frac{2\sqrt{\pi}}{e}\left(\frac{2(k+1)}{e}\right)^{k+1}$$

and consequently

$$\log n \geq (k+1)\left(\log(k+1) + \log\left(\frac{2}{e}\right)\right) + \log\left(\frac{2\sqrt{\pi}}{e}\right) \geq (k+1)\big(\log(k+1) - 1\big).$$

As $x \geq y > 1$ implies $\frac{x}{\log x} \geq \frac{y}{\log y}$, we have

$$\frac{\log n}{\log\log n} \geq \frac{(k+1)\big(\log(k+1)-1\big)}{\log(k+1) + \log\big(\log(k+1)-1\big)}$$

and it follows that $k = O(\log n/\log\log n)$.
And finally, as [7] exhibits some cographs having linearity $\Omega(\log n/\log\log n)$, consequently, the upper bound provided by the lemma is tight. □

We now prove the main result aimed by this paper: linearity is a strictly more powerful encoding than contiguity, which means, from Remark 3, that there exists some graph families for which the linearity is asymptotically negligible in front of the contiguity (hereafter denoted $cont(G)$ for a graph G).

Corollary 1 *There exists a series of graphs G_h, $h \geq 1$, such that $cont(G_h)/lin(G_h)$ tends to infnity when h tends to infinity.*

Proof. For $h \geq 1$, let G_h be the connected cograph whose cotree is a complete binary tree of height h and let $n = 2^h$ denote the number of vertices of G_h. It is proven in [7] that $cont(G_h) = \Theta(\log n)$ and that $lin(G_h) = \Omega(\log n/\log\log n)$. Then, Theorem 1 above implies that $lin(G_h) = \Theta(\log n/\log\log n)$ and therefore $cont(G_h)/lin(G_h) = \Theta(\log\log n)$, which achieves the proof. □

5 Perspectives

In this paper, we showed that linearity provides a strictly more powerful encoding for graphs than contiguity does, meaning that the ratio between the contiguity and the linearity of a graph is not bounded by a constant. From a practical point of view, the meaning of our result is that using several orders, instead of just one, for grouping neighbourhoods of vertices can greatly enhance compression rates in some cases.

We obtained this result by exhibiting a graph family, namely a subfamily of cographs, for which the ratio between the contiguity and the linearity tends to infinity as fast as $\Omega(\log\log n)$, with n the number of vertices in the graph. As a by-product of our proof, but meaningful in itself, we also showed tight bounds for the worst-case linearity of cographs on n vertices; tight bounds were previously known for contiguity. Several questions naturally arises from these results and others.

Open question 1 *What is the worst case contiguity and the worst-case linearity of arbitrary graphs?*

It is straightforward to see that both of these values are bounded by $n/2$. Conversely, since there are $2^{n(n-1)/2}$ graphs on n labelled vertices and since contiguity and linearity do not depend on the labels of the vertices, then both encodings must use at least n^2 bits for graphs on n vertices. Moreover, when the value of the parameter is k, the size of the corresponding encoding is $O(k\,n)$ integers, that is $O(k\,n\log n)$ bits. Consequently, both parameters must be at least $\Omega(n/\log n)$ in the worst case. For contiguity, [8] gave an upper bound asymptotically equivalent to $n/4$. Is $\Omega(n)$ indeed the worst-case contiguity of a graph? Is the worst-case for linearity the same as the one for contiguity? Another appealing question which is closely related is the following.

Open question 2 *For arbitrary graphs, what is the maximum gap between contiguity and linearity?*

In other words, let $(G_n)_{n\geq 1}$ be a family of graphs on n vertices and let $f(n) = cont(G_n)/lin(G_n)$. Can $f(n)$ tends to infinity faster than $\Omega(\log\log n)$? What is the maximum asymptotic growth possible for $f(n)$? Answering those questions would be both theoretically and practically of key interest for the field of graph encoding.

References

[1] Alberto Apostolico and Guido Drovandi. Graph compression by BFS. *Algorithms*, 2(3):1031–1044, 2009.

[2] P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *WWW'04*, pages 595–602. ACM, 2004.

[3] P. Boldi and S. Vigna. Codes for the world wide web. *Internet Mathematics*, 2(4):407–429, 2005.

[4] Paolo Boldi, Massimo Santini, and Sebastiano Vigna. Permuting web and social graphs. *Internet Mathematics*, 6(3):257–283, 2009.

[5] D.G. Corneil, H. Lerchs, and L.Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.

[6] Christophe Crespelle and Philippe Gambette. Efficient neighbourhood encoding for interval graphs and permutation graphs and $O(n)$ breadth-first search. In *IWOCA'09*, number 5874 in LNCS, pages 146–157, 2009.

[7] Christophe Crespelle and Philippe Gambette. (nearly-)tight bounds on the contiguity and linearity of cographs. *Theoretical Computer Science*, 522:1–12, 2014.

[8] C. Gavoille and D. Peleg. The compactness of interval routing. *SIAM Journal on Discrete Mathematics*, 12(4):459–473, 1999.

[9] P.W. Goldberg, M.C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[10] L. Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006.

[11] Hossein Maserrat and Jian Pei. Neighbor query friendly compression of social networks. In *KDD'10*, pages 533–542. ACM, 2010.

[12] F.S. Roberts. *Representations of Indifference Relations*. PhD thesis, Stanford University, 1968.

[13] G. Turan. On the succinct representation of graphs. *Discr. Appl. Math.*, 8:289–294, 1984.

[14] R. Wang, F.C.M. Lau, and Y. Zhao. Hamiltonicity of regular graphs and blocks of consecutive ones in symmetric matrices. *Discr. Appl. Math.*, 155(17):2312–2320, 2007.