

Maximising the quantity of valid 2-paths on cographs

Gustavo M. Leal, Humberto J. Longo, Hebert C. Silva, Les R. Foulds

Instituto de Informática

Universidade Federal de Goiás

Alameda Palmeiras, Quadra D, Campus Samambaia

CEP 74001-970, Goiânia, Goiás, Brasil

`gustavoleal@discente.ufg.br`

`{lesfoulds, longo, hebert}@ufg.br`

ABSTRACT

Assigning labels to the vertices and/or edges of a given graph $G = (V, E)$, respecting predefined conditions, is a well-known research problem in the field of Graph Theory. The labelling of a graph G , of order n , is defined as a *numbering* when the set of integers $\{1, \dots, n\}$ is used to label $V(G)$ in a distinct way. A path with three vertices (2-path) is termed *valid* if the label associated with its central vertex is smaller than the labels associated with the endpoints of the path. The *Path Validity Problem* involves finding a numbering that optimises the quantity of valid 2-paths in G . The focus of this work is on the class of cographs, presenting an polynomial algorithm based on dynamic programming which maximizes the quantity of valid 2-paths in cographs. Also some interesting properties related to the problem are presented.

KEYWORDS. Numbering, 2-path, valid path, cograph, dynamic programming.

1. Introduction

The subfield of graph theory that deals with graph labelling was introduced by Rosa [1967]. Essentially, labelling a graph is the assignment of labels to its sets of vertices or edges (or both), subject to some predefined constraints. Since Rosa's seminal paper, thousands of studies have been published, and most of them are listed in Gallian's Dynamic Survey [Gallian, 2022], which has been updated for over 25 years.

Several well-known graph optimisation problems can be formulated in terms of labelling as follows: given a graph G , find a labelling of G such that a mathematical function of some measure associated with the labelling is optimized. Applications fields include, among many others, network addressing [Judd and Beer, 1995], circuit layout code design [Shieh et al., 2011] and the bandwidth, minimum-sum and cut-width problems [Chung, 1981; Juvan and Mohar, 1992].

Some problems in this field are also related to defining some order for the vertices and applying some set of operations based on this order. This kind of labelling can be formally defined as numbering, where each vertex is labelled with a distinct number of the set $\{1, \dots, n\}$ (in this work n always refers to the quantity of vertices in the graph and this type of labelling will be referred as *vertex numbering*). Such labelling can also be seen as a permutation of the vertices.

Bloom and Golomb [1977] showed that vertex numbering has a wide range of applications, most of them based on mathematical models, such as the analysis of various coding theory problems, including the design of good radar-type codes, sync-set and convolutional codes with optimal autocorrelation properties.

Foulds and Longo [2023] introduced the concept of *path validity*, which is based on the vertex numbering of a graph, i.e., a path is valid if it is a 2-path (a path with three vertices) and the label of the middle vertex is smaller than the labels of its endpoints (note that this concept is defined

only for 2-paths). The *Path Validity Problem* involves finding a numbering for a given graph G that optimizes the number of valid paths in G . Leal [2024] demonstrated that the maximization variant of the path validity problem is $\mathcal{NP}$ -hard, even when the problem is restricted to cubic graphs.

This problem arises, for example, in the enumeration of chordless cycles in graphs, which has application in the prediction of nuclear magnetic resonance, chemical shift values, and ecological networks [Dias et al., 2013]. Furthermore, the valid path problem has other applications related to numbered graphs, such as in discrete system models in which only partial comparisons of the vertex numbers can be made.

Another possible application can be found in systems with intermediate validation. For instance, suppose that a communication network where each node has a different rank and communication between two devices at nodes a and b must be validated/mediated by a device at node c , connected to both a and b , with a higher rank than those of a and b . Thus, the quantity of possible connections, subject to intermediate validation, is equivalent to the number of *valid paths*. Furthermore, ranking the entire network is equivalent to numbering all its nodes and maximising the number of valid connections is equivalent to maximising the quantity of *valid paths* in the support graph of the network.

In this work, this numbering problem is studied, focusing on the complexity of the maximisation variant of the problem when restrict to cographs. The remaining sections are organized as follows. Section 2 provides the formal definitions used throughout the text. Section 3 presents some results valid for graphs in general, and section 4 presents results related to cographs. Section 5 presents an algorithm, with asymptotic polynomial time complexity $O(n^3)$, to find a numbering that induces the maximum number of valid paths in cographs. This algorithm uses dynamic programming techniques combined with the structure of parse trees. Finally, section 6 presents a conclusion of the work.

2. Definitions

Let $G = (V(G), E(G))$ be a finite, undirected, simple graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $Adj(u) = \{v \in V(G) \mid \{u, v\} \in E(G)\}$. A *numbering* π of an undirected graph G is a bijective mapping from $\{1, 2, \dots, n\}$ to $V(G)$, where $\pi(v)$ denotes the label of vertex $v \in V(G)$. A 2-path $\langle x, u, y \rangle$ in a numbered graph G is said to be *valid* if the number associated with its middle vertex u is smaller than those of its endpoints x and y , i.e., $\pi(u) < \min\{\pi(x), \pi(y)\}$. The validity $\phi_\pi(G)$, of a numbering π of G , is defined as the quantity of valid 2-paths of G induced by π , i.e.,

$$\phi_\pi(G) = |\{\langle x, u, y \rangle \mid u \in V(G), x, y \in Adj(u) \text{ and } \pi(u) < \min\{\pi(x), \pi(y)\}\}|. \quad (1)$$

The *maximum validity* $\phi_{\max}(G)$ of G is the maximum of $\phi_\pi(G)$ over all possible numberings of G , i.e.,

$$\phi_{\max}(G) = \max\{\phi_\pi(G) \mid \pi \text{ is a numbering of } G\}. \quad (2)$$

Given a numbering π , the quantity of valid paths induced by a particular vertex u , where u is the middle vertex of a valid path, is defined as:

$$\varphi_\pi(u) = |\{[x, y] \mid x, y \in Adj(u) \text{ and } \pi(u) < \min\{\pi(x), \pi(y)\}\}| = \binom{\psi_\pi(u)}{2}, \quad (3)$$

where

$$\psi_\pi(u) = |\{x \mid x \in Adj(u) \text{ and } \pi(u) < \pi(x)\}| \quad (4)$$

is the cardinality of the subset of vertices adjacent to u that have labels smaller than that of u .

For general graphs, $\phi_{\min}(G)$ and $\phi_{\max}(G)$ can vary significantly, depending on the graph structure and how the numbering is carried out. If no constraints are imposed, then $n!$ different numberings are possible, inducing sets of valid paths that may vary quite markedly. Figure 1 illustrates this phenomenon for three distinct numberings of the C_6 graph, with the valid paths indicated in red. The leftmost numbering induces only a single valid path, the minimum quantity, namely $\langle 2, 1, 6 \rangle$, and is a manifestation of $\phi_{\min}(C_6)$. The second numbering induces two valid paths, namely $\langle 4, 1, 5 \rangle$ and $\langle 3, 2, 5 \rangle$, and is neither minimal nor maximal. The rightmost numbering induces the maximal quantity, a triple, of valid paths, namely $\langle 4, 1, 6 \rangle$, $\langle 4, 2, 5 \rangle$ and $\langle 5, 3, 6 \rangle$, and is a manifestation of $\phi_{\max}(C_6)$.

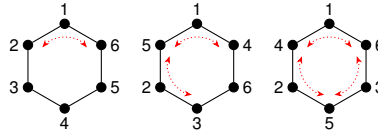


Figure 1: Three numberings of the C_6 graph.

3. General results

Lemma 3.1, given below, introduces an useful numbering property that is used later in the proofs of Lemmas 3.2 and 4.1, stated later. Given a graph G , Lemma 3.2 states that there is a numbering π that induces $\phi_{\max}(G)$ and the quantity of valid paths induced by each of the vertices in G is non-decreasing, related to the order of vertices established by the numbering itself. This result establishes a strong property in the structure of numberings which induces $\phi_{\max}(G)$, leading to possible applications in multiple classes of graphs, as presented in Section 4.

Lemma 3.1 *Given a graph G and $u, v \in V(G)$, let π be a numbering such that $\phi_{\pi}(G) = \phi_{\max}(G)$. If $(u, v) \in E(G)$ and $\pi(u) = \pi(v) - 1$ then $\varphi_{\pi}(u) > \varphi_{\pi}(v)$*

Proof. $\phi_{\pi}(G)$ is maximum, so no change in the labels can increase the number of valid paths. The swap of labels of u and v leads to a numbering α changing $\psi_{\pi}(u)$ to $\psi_{\alpha}(u) = \psi_{\pi}(u) - 1$ and $\psi_{\pi}(v)$ to $\psi_{\alpha}(v) = \psi_{\pi}(v) + 1$, respectively. The quantity of valid paths induced by any other vertex will remain the same, but $\varphi_{\alpha}(u)$ and $\varphi_{\alpha}(v)$ would change from $\binom{\psi_{\pi}(u)}{2}$ and $\binom{\psi_{\pi}(v)}{2}$ to $\binom{\psi_{\pi}(u)-1}{2}$ and $\binom{\psi_{\pi}(v)+1}{2}$

Since $\phi_{\pi}(G) = \phi_{\max}(G)$ thus $\phi_{\pi}(G) \geq \phi_{\alpha}(G)$. Given that the quantity of valid paths induced by any vertex other than u and v remains the same in both numberings, the relation $\binom{\psi_{\pi}(u)}{2} + \binom{\psi_{\pi}(v)}{2} \geq \binom{\psi_{\pi}(u)-1}{2} + \binom{\psi_{\pi}(v)+1}{2}$ must be true. Therefore:

$$\binom{\psi_{\pi}(u)}{2} + \binom{\psi_{\pi}(v)}{2} \geq \binom{\psi_{\pi}(u)-1}{2} + \binom{\psi_{\pi}(v)+1}{2}, \quad (5)$$

$$\binom{\psi_{\pi}(u)}{2} + \binom{\psi_{\pi}(v)}{2} - \binom{\psi_{\pi}(u)-1}{2} - \binom{\psi_{\pi}(v)+1}{2} \geq 0,$$

$$2 \cdot \psi_{\pi}(u) - 2 \cdot \psi_{\pi}(v) - 2 \geq 0,$$

$$2 \cdot \psi_{\pi}(u) \geq 2 \cdot \psi_{\pi}(v) + 2,$$

$$\psi_{\pi}(u) \geq \psi_{\pi}(v) + 1,$$

$$\psi_{\pi}(u) > \psi_{\pi}(v). \quad (6)$$

□

Lemma 3.2 Given a graph G , there is a numbering π such that $\phi_\pi(G) = \phi_{max}(G)$ and if $\pi(u) < \pi(v)$ then $\varphi_\pi(u) \geq \varphi_\pi(v)$.

Proof. Suppose that there is a numbering α such that $\phi_\alpha(G) = \phi_{max}(G)$ and there is two vertices u' and v' , where $\alpha(u') < \alpha(v')$ and $\varphi_\alpha(u') < \varphi_\alpha(v')$. The following proof presents a transformation from α to π . It is sufficient to analyse only vertices $u, v \in V(G)$ with adjacent labels, i.e., $\alpha(u) = \alpha(v) - 1$. There are only two possibilities, stated as:

- i) $(u, v) \in E(G)$: By Lemma 3.1, $\varphi_\alpha(u) \geq \varphi_\alpha(v)$. Thus, $\pi(u) = \alpha(u)$.
- ii) $(u, v) \notin E(G)$: Swapping the labels of u and v in α will not affect $\phi_\alpha(G)$. Thus:
 - if $\varphi_\alpha(u) \geq \varphi_\alpha(v)$, label $\pi(u)$ as $\alpha(u)$;
 - otherwise, label $\pi(v)$ as $\alpha(u)$.

□

It is important to note that Lemma 3.2 does not implies a property over all numberings which induces $\phi_{max}(G)$, it proposes that there is a numbering with the characteristics presented in the lemma itself.

4. Cographs

Cographs, or complement-reducible graphs, are a class of graphs that can be constructed using a recursive process involving disjoint unions and joins, starting from single-vertex graphs. *Parse Trees* are tree representations of cographs, where each internal node corresponds to an operation (union or join), and leaves represent individual vertices. This hierarchical structure facilitates the efficient representation and processing of cographs, making them valuable for various combinatorial and computational applications. A parse tree of a cograph is a special type of modular decomposition, i.e. a unique decomposition of the vertices into nested modules. Dahlhaus et al. [2001] proposed an algorithm that performs this construction in linear time complexity.

Formally, a finite, undirected, simple graph $G = (V(G), E(G))$ is called a *cograph* (or *complement-reducible graph*) if it can be constructed from a single vertex by a sequence of the following operations: (i) disjoint union of two pre-existing cographs; or (ii) join of two pre-existing cographs; The notations $+$ and $\cup$ will be used to denote the join and the disjoint union operations, respectively. The *parse tree* data structure will be denoted as $Pt(G)$.

The join operation of two graphs G and H , starts with the disjoint union of G and H and then add an edge between every pair (u, v) of vertices, such that $u \in V(G)$ and $v \in V(H)$. Formally $V(G + H) = V(G) \cup V(H)$ and $E(G + H) = E(G) \cup E(H) \cup \{(u, v) \mid u \in V(G), v \in V(H)\}$. Figure 2(c) illustrates an instance of a cograph $G + H$, build from the cographs G and H shown in (a) and (b), and its respective parse tree $Pt(G + H)$.

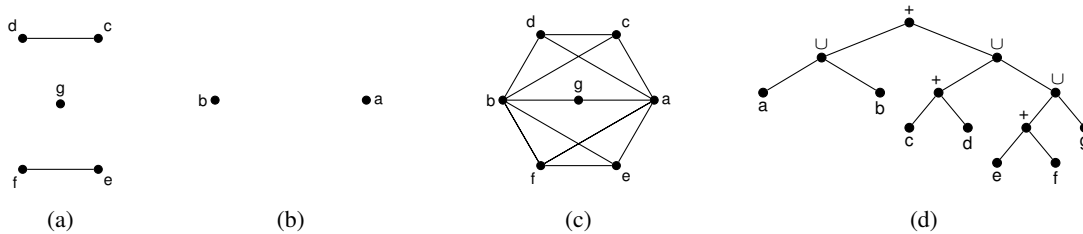


Figure 2: (a) cograph G , (b) cograph H , (c) cograph $G + H$ and (d) parse tree $Pt(G + H)$.

Lemma 4.1, given below, states that the numberings that induce ϕ_{max} in two cographs G and H are maintained as partial orders of the numbering that induce ϕ_{max} in the cograph $G + H$.

Lemma 4.1 *Let G and H be two cographs and π be a numbering of G satisfying Lemma 3.2. There is a numbering α such that $\phi_\alpha(G + H) = \phi_{max}(G + H)$ and the relative order of the vertices in G is equivalent to their order in π , i.e.,*

Proof. Any pair $u, v \in V(G)$, such that $\pi(u) = \pi(v) - 1$, will have only vertices from H with labels between them in α . Thus, suppose that there are q vertices of H with labels greater than $\alpha(v)$ and p vertices of h with labels greater than $\alpha(u)$. Since $\alpha(u) < \alpha(v)$, $p \geq q$. The quantity of valid paths induced by u and v in α will be, respectively, $\varphi_\alpha(u) = \binom{\psi_\pi(u)+p}{2}$ and $\varphi_\alpha(v) = \binom{\psi_\pi(v)+q}{2}$. If labels $\alpha(u)$ and $\alpha(v)$ are swapped, generating a new numbering α' , there are two possibilities:

- i) $\{u, v\} \notin E(G)$: the quantity of valid paths induced by α' would be $\varphi_{\alpha'}(u) = \binom{\psi_\pi(u)+q}{2}$ and $\varphi_{\alpha'}(v) = \binom{\psi_\pi(v)+p}{2}$. Since $\phi_\alpha(G + H) = \phi_{max}(G + H)$:

$$\binom{\psi_\pi(u)+p}{2} + \binom{\psi_\pi(v)+q}{2} \geq \binom{\psi_\pi(u)+q}{2} + \binom{\psi_\pi(v)+p}{2}, \quad (7)$$

$$\begin{aligned} \binom{\psi_\pi(u)+p}{2} + \binom{\psi_\pi(v)+q}{2} - \binom{\psi_\pi(u)+q}{2} - \binom{\psi_\pi(v)+p}{2} &\geq 0, \\ 2 \cdot \psi_\pi(u) \cdot (p - q) - 2 \cdot \psi_\pi(v) \cdot (p - q) &\geq 0, \\ \psi_\pi(u) \cdot (p - q) - \psi_\pi(v) \cdot (p - q) &\geq 0, \\ \psi_\pi(u) \cdot (p - q) &\geq \psi_\pi(v) \cdot (p - q). \end{aligned} \quad (8)$$

Since $p \geq q$ and, by Lemma 3.2, $\psi_\pi(u) \geq \psi_\pi(v)$, it follows that the inequality (8) is true.

- ii) $\{u, v\} \in E(G)$: the quantity of valid paths induced by α' would be $\varphi_{\alpha'}(u) = \binom{\psi_\pi(u)+q-1}{2}$ and $\varphi_{\alpha'}(v) = \binom{\psi_\pi(v)+p+1}{2}$. Thus,

$$\binom{\psi_\pi(u)+p}{2} + \binom{\psi_\pi(v)+q}{2} \geq \binom{\psi_\pi(u)+q-1}{2} + \binom{\psi_\pi(v)+p+1}{2}, \quad (9)$$

$$\begin{aligned} \binom{\psi_\pi(u)+p}{2} + \binom{\psi_\pi(v)+q}{2} - \binom{\psi_\pi(u)+q-1}{2} - \binom{\psi_\pi(v)+p+1}{2} &\geq 0, \\ 2 \cdot \psi_\pi(u) \cdot (p - q + 1) - 2 \cdot \psi_\pi(v) \cdot (p - q - 1) - 2 \cdot p + 2 \cdot q - 2 &\geq 0, \\ \psi_\pi(u) \cdot (p - q + 1) - \psi_\pi(v) \cdot (p - q - 1) - p + q - 1 &\geq 0, \\ \psi_\pi(u) \cdot (p - q + 1) + q &\geq \psi_\pi(v) \cdot (p - q - 1) + p + 1, \\ \psi_\pi(u) \cdot (p - q) + q + \psi_\pi(u) + \psi_\pi(v) &\geq \psi_\pi(v) \cdot (p - q) + p + 1. \end{aligned}$$

Since $p \geq q$ and, by Lemma 3.1, $\psi_\pi(u) > \psi_\pi(v)$, the worst case scenario would be when $\psi_\pi(u) = \psi_\pi(v) + 1$. Therefore:

$$\begin{aligned} (\psi_\pi(v) + 1) \cdot (p - q) + q + \psi_\pi(v) + \psi_\pi(v) + 1 &\geq \psi_\pi(v) \cdot (p - q) + p + 1, \\ \psi_\pi(v) \cdot (p - q) + p + \psi_\pi(v) + \psi_\pi(v) + 1 &\geq \psi_\pi(v) \cdot (p - q) + p + 1, \\ 2 \cdot \psi_\pi(v) &\geq 0. \end{aligned} \quad (10)$$

The inequality (10) is also true, which concludes the proof.

□

The property stated by Lemma 4.1 leads to a recurrence function (Definition 4.2), which allows obtaining $\phi_{max}(G + H)$ if the numberings $\phi_{max}(G)$ and $\phi_{max}(H)$ are known in advance.

Definition 4.2 Given two cographs G and H , let π_1 and π_2 be two numberings associated with G and H , respectively, both satisfying Lemma 3.2 and such that $\phi_{\pi_1} = \phi_{max}(G)$ and $\phi_{\pi_2} = \phi_{max}(H)$. Let $\langle v_1, \dots, v_n \rangle$ and $\langle u_1, \dots, u_n \rangle$ be permutations of $V(G)$ and $V(H)$, respectively, such that if $k < l$ then $\psi_{\pi_1}(v_k) \geq \psi_{\pi_1}(v_l)$ and $\psi_{\pi_2}(u_k) \geq \psi_{\pi_2}(u_l)$. The recurrence function $\Gamma(i, j)$ is defined as follows:

$$\Gamma(i, j) = \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0; \\ \binom{\psi_{\pi_1}(v_i)}{2} + \Gamma(i - 1, j), & \text{if } i > 0 \text{ and } j = 0; \\ \binom{\psi_{\pi_2}(u_j)}{2} + \Gamma(i, j - 1), & \text{if } i = 0 \text{ and } j > 0; \\ \max\{\binom{\psi_{\pi_1}(v_i) + j}{2} + \Gamma(i - 1, j), \binom{\psi_{\pi_2}(u_j) + i}{2} + \Gamma(i, j - 1)\}, & \text{otherwise.} \end{cases}$$

This recurrence will be further used in Algorithm 1 (*JoinDP()*), whose correctness proof is exactly the one proposed for Lemma 4.3, given next. This algorithm is subsequently used as one of the steps of Algorithm 2 (*CoTreeMax()*), which obtains $\phi_{max}(G)$ for any cograph G .

Lemma 4.3 $\Gamma(i, j)$ is equivalent to the maximum sum of $\varphi_\alpha(x)$, for every vertex $x \in S = \{v_i, v_{i-1}, \dots, v_1\} \cup \{u_j, u_{j-1}, \dots, u_1\}$, of some numbering α of $G + H$ such that if $x \notin S$ and $y \in S$, then $\alpha(y) > \alpha(x)$.

Proof. By Lemma 4.1, there is a numbering α such that $\phi_\alpha(G + H) = \phi_{max}(G + H)$ and the relative order of π_1 and π_2 is maintained in relation to α . Given this fact, vertex in S that has the smaller label in α must be u_j or v_i . Thus, a proof by induction can be established.

The base case is $i = 0$ or $j = 0$. $\Gamma(i, 0)$ is equivalent to the sum of $\varphi_\alpha(x)$ of some numbering α of $G + H$, for every vertex $x \in S = \{v_i, v_{i-1}, \dots, v_1\}$, such that $\sum_{k=1}^i \binom{\alpha(v_k)}{2}$ is maximised and every other vertex belonging to $G + H$ have a smaller label than the vertices in S . By Lemma 4.1, the option that maximises this summation is to assign to v_i the smallest possible label, maintaining $\psi_{\pi_1}(v_i)$, leading to the sum $\binom{\psi_{\pi_1}(v_i)}{2} + \Gamma(i - 1, 0)$. The proof of the case $\Gamma(0, j)$ is similar.

Suppose now that the function $\Gamma(i', j')$ is correctly defined for all pairs (i', j') such that $0 \leq i' \leq i, 0 \leq j' \leq j$ and either i' or j' is strictly smaller than $\min\{i, j\}$.

The same relationship as in the base case can be analysed. Based on Lemma 4.1, it is possible to choose vertex u_j or vertex v_i to receive the smaller label. If u_j is chosen, $\psi_\alpha(u_j) = \psi_{\pi_2}(u_j) + i$ since every vertex of G is adjacent to u_j . Analogously, if v_i is chosen, $\psi_\alpha(v_i) = \psi_{\pi_1}(v_i) + j$. Therefore, $\Gamma(i, j)$ will be the maximum of both options, which is exactly the general step of the recurrence. Given that the proposition, by induction hypothesis, is valid for $\Gamma(i - 1, j)$ and $\Gamma(i, j - 1)$, follows that it is also valid for $\Gamma(i, j)$.

□

Corollary 4.4 $\Gamma(|V(G)|, |V(H)|) = \phi_{max}(G + H)$

Proof. $\Gamma(|V(G)|, |V(H)|)$ is a particular case of Lemma 4.3.

□

5. Algorithms

This section presents an algorithm which returns $\phi_{max}(G)$ when G is a cograph. The algorithm traverse the Parse Tree of G , generating a numbering for each subtree and merging the numbering of both subtrees of a vertex in a parse tree.

Algorithm 1: JoinDP(V_1, V_2)

Input: Two unidimensional matrices V_1 and V_2 of sizes n_1 and n_2 , respectively.
Output: Merge of V_1 and V_2

```

1   $memo[0 \dots n_1, 0 \dots n_2] \leftarrow 0$ 
2   $V[1 \dots n_1 + n_2] \leftarrow 0$ 
3  for  $i \leftarrow n_1$  to 0 do
4      for  $j \leftarrow n_2$  to 0 do
5          if  $i = n_1$  and  $j = n_2$  then
6               $memo[i, j] \leftarrow 0$ ;
7          else if  $i = n_1$  then
8               $memo[i, j] \leftarrow \binom{V_2[j+1]+n_1-i}{2} + memo[i, j+1]$ 
9          else if  $j = n_2$  then
10              $memo[i, j] \leftarrow \binom{V_1[i+1]+n_2-j}{2} + memo[i+1, j]$ 
11         else
12              $memo[i, j] \leftarrow$ 
13                  $\max(\binom{V_1[i+1]+n_2-j}{2} + memo[i+1, j], \binom{V_2[j+1]+n_1-i}{2} + memo[i, j+1])$ 
14  $i, j \leftarrow 0$ ;
15 while  $(i + j \neq n_1 + n_2)$  do
16     if  $i = n_1$  then
17          $V[i + j] \leftarrow V_2[j] + n_1 - i$ 
18          $j \leftarrow j + 1$ 
19     else if  $j = n_2$  then
20          $V[i + j] \leftarrow V_1[i] + n_2 - j$ 
21          $i \leftarrow i + 1$ 
22     else if  $memo[i, j] = \binom{V_1[i]+n_2-j}{2} + memo[i+1, j]$  then
23          $V[i + j] \leftarrow V_1[i] + n_2 - j$ 
24          $i \leftarrow i + 1$ 
25     else
26          $V[i + j] \leftarrow V_2[j] + n_1 - i$ 
27          $j \leftarrow j + 1$ 
28 return  $V$ 
```

When merging cographs using a parse tree, the join operation adds multiple edges to each vertex of the graph, which can significantly increase the number of valid paths. The algorithm presented below maximizes the number of valid paths by performing this operation based on the Definition 4.2. The function $\Gamma(i, j)$ would require an exponential time complexity to be evaluated. However, with the bottom-up Dynamic Programming approach presented in the algorithm, each substate is computed only once, which reduces the asymptotic time complexity to just $O(n^2)$.

Algorithm 2: $ParseTreeMax(Pt(G))$

Input: A Parse Tree $Pt(G)$ for G .

Output: A Vector V representing the list of ψ 's for the vertices from G which leads to $\phi_{max}(G)$.

```

1 if  $|Pt(G)| = 1$  then return  $\{0\}$  ;
2  $V_1 \leftarrow ParseTreeMax$  (Parse Tree starting at left child of the root of  $Pt(G)$ );
3  $V_2 \leftarrow ParseTreeMax$  (Parse Tree starting at right child of the root of  $Pt(G)$ );
4 if (operation at root of  $Pt(G)$  is  $+$ ) then  $V \leftarrow JoinDP(V_1, V_2)$  else  $V \leftarrow Merge(V_1, V_2)$  ;
5 return  $V$ .

```

Algorithm 1 ($JoinDP()$) uses a dynamic programming approach to join two parse trees. The two unidimensional matrices V_1 and V_2 , input to the algorithm, represent the ψ of the root vertices of each parse tree. The bidimensional matrix $memo[i, j]$ is equivalent to the recurrence in the Definition 4.2, steps from 1 to 11 perform the calculation of the recursive function and steps from 14 to 24 are the traverse in the matrix $memo$ to obtain the matrix equivalent to the ψ of $G + H$ cograph.

Algorithm 2 ($CoTreeMax$) traverses the structure of a parse tree. Each subtree of a parse tree is also a parse tree, the algorithm takes advantage of this feature to use the subtree solution in the current parse tree. The return of the algorithm is an matrix representing ψ of each vertex defined in equation (4).

Given a binary parse tree $Pt(G)$, step 1 of Algorithm 2 deals with the case that $Pt(G)$ has only one vertex. Clearly, ψ of that vertex will be 0. Otherwise, steps 2 and 3 recursively process both subtrees of $Pt(G)$. Next, if the operation at this level of $Pt(G)$ is a join, at step 4 Algorithm 1, $JoinDP()$ routine, is called to performs the merge using a dynamic programming approach. Otherwise, if the merge consists of a union operation, theoretically any numbering that respects partial orderings would be feasible. However, to maintain the required property of Lemma 3.2, the union operation is performed by the routine $Merge()$, which consists of a simple loop. Since V_1 and V_2 are already sorted, the construction of V can be done linearly, iterating over both of them.

Algorithm 1 ($JoinDP()$) has asymptotic time complexity of $O(n^2)$, and is called at least n times. Thus, the time complexity of Algorithm 2 ($CoTreeMax()$) is $O(n^3)$. This algorithm computes only $\phi_{max}(G)$, but an extension that also returns the numbering of each vertex can be easily derived.

Lemma 5.1 shows a proof that the Algorithm 2 leads to a numbering which induces $\phi_{max}(G)$ for a given cograph G , concluding the correctness of the algorithm.

Lemma 5.1 *Given a cograph G , there is a numbering α such that the set $S = \bigcup_{u \in V(G)} \psi_\alpha(u)$ is equal to $ParseTreeMax(Pt(G))$ and $\phi_\alpha(G) = \phi_{max}(G)$.*

Proof. The proof is by induction in the cardinality of $V(G)$. Thus, when $|V(G)| = 1$ the quantity of valid paths in graphs with only one vertex is 0, since $ParseTreeMax(Pt(G)) = \{0\}$, and the property is trivially true.

Suppose that the function $ParseTreeMax(Pt(G'))$ is correctly defined for all G' such that $|V(G')| < |V(G)|$. By hypothesis, both V_1 and V_2 will represent a list of ψ 's which induces $\phi_{max}(G')$ and $\phi_{max}(H)$, respectively.

Algorithm 1 ($JoinDP$) also return a list of ψ 's which maintain this property, based on Corollary 4.4. Thus, if the last operation is a join operation, the algorithm maintains the property if V_1 and V_2 maintain the property from Lemma 3.2. However, it is clear that $ParseTreeMax$

always returns an ordered list of ψ 's. If the last operation is a disjoint union, the list of ψ 's remains unchanged from V_1 and V_2 . Consequently, any returned list preserves the property. □

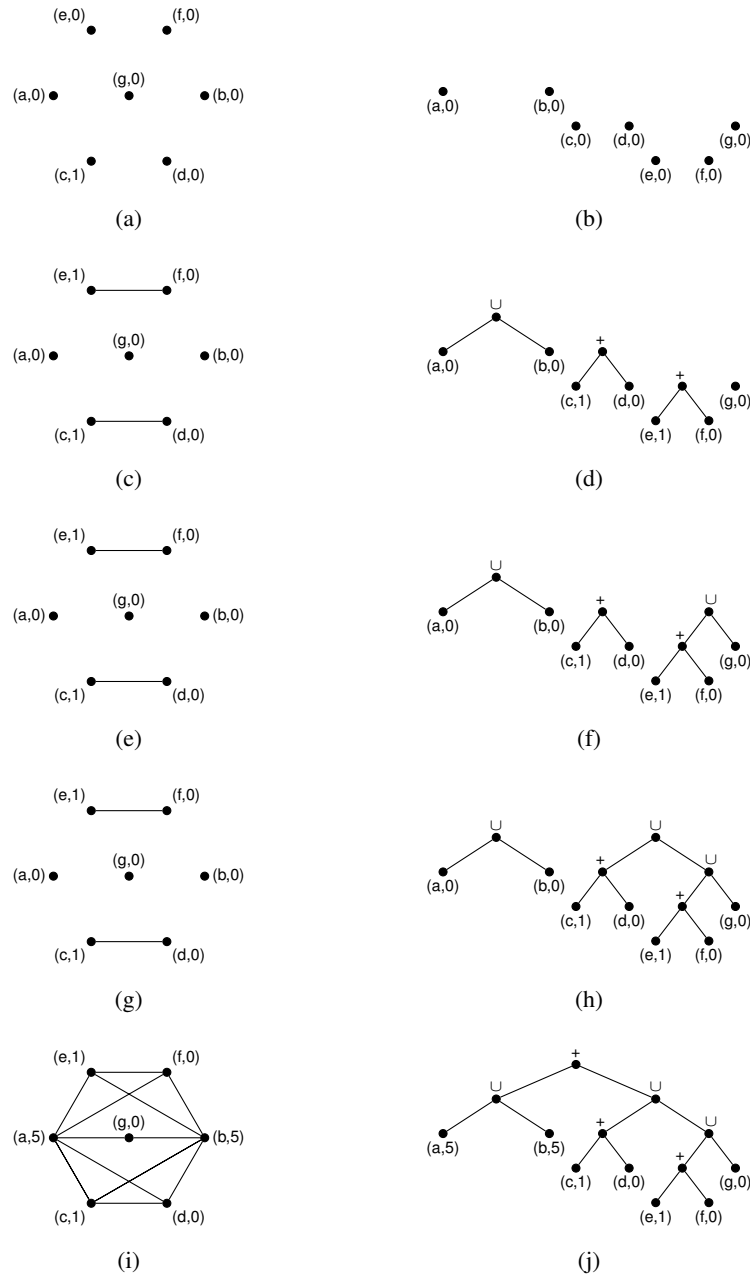


Figure 3: Cographs and parse trees: (a) and (b) step 1, (c) and (d) step 2, (e) and (f) step 3, (g) and (h) step 4, (i) and (j) step 5 of Algorithm 2.

Figure 3 illustrates the steps of the execution of Algorithm 2, receiving as input the cograph G shown in Figure 2. The algorithm follows a divide and conquer approach, recursively

processing each subtree of $Pt(G)$. Subfigures 3(a) and 3(b) show the cograph and the parse tree at the beginning of the first iteration of the algorithm. Each vertex has a label with a pair: an identification label represented by a letter and its respective ψ at this stage of the algorithm. At the leaves of each $Pt(G)$ (subfigures 3(b), 3(d), 3(f), 3(h) and 3(j)), the ψ of each vertex will be 0, as specified in step 1 of the Algorithm 2.

Along the first iteration, there is a disjoint union of the vertices a and b , a join union of vertices d and c and of vertices e and f , as shown in 3(c) and 3(d). After the disjoint union operation, the set of ψ 's is only merged, thus they remains the same. It is clear that a join of two vector of the form $\{0\}$ will lead to a vector of the form $\{1, 0\}$ which is exactly what happens in the join of c and d and in the join of e and f . Afterwards, there is a disjoint union of the subtree with nodes e and f and the subtree with node g (subfigures 3(e) and 3(f)). The next step is also a disjoint union of two sets of vertices, $\{c, d\}$ and $\{e, f, g\}$. Every ψ remains the same (subfigures 3(g) and 3(h)).

After these steps, the vectors representing the ψ 's are $\{(a, 0), (b, 0)\}$ and $\{(c, 1), (e, 1), (d, 0), (f, 0), (g, 0)\}$. It is important to point out that the algorithm does not maintain a label in the vector, it is only to help the reader. the Matrix *memo* constructed by the Algorithm 1 and the path for the optimal solution are presented in Figure 4(a). The arrows in the figure show the path in the optimal solution, leading to the following vector of ψ 's: $\{(a, 5), (b, 5), (c, 1), (e, 1), (d, 0), (f, 0), (g, 0)\}$. The final Step leads to the cograph and the parse tree shown in 3(i) and 3(j), concluding that the maximum quantity of valid paths is 20. A possible numbering that would lead to this list of ψ 's is shown in Figure 4(b).

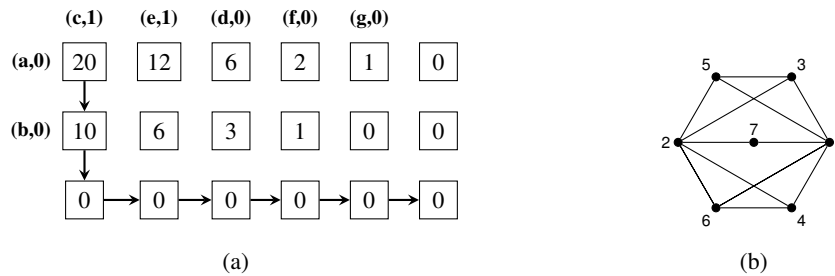


Figure 4: (a) *memo* matrix in Step 5 of Algorithm 2 and (b) Cograph G with numbering that leads to $\phi_{max}(G)$.

6. Conclusion

The *Path Validity Problem* involves finding a numbering that optimizes the number of valid paths in a given graph. Since this problem belongs to the $\mathcal{NP}$ -hard class for graphs in general, studying it in more restricted classes of graphs improves the knowledge about the problem and its properties.

Thus, this work investigated the behaviour of this numbering problem when restricted to the class of cographs. Some interesting properties related to the structure of the maximization variant of the problem were observed and stated as the Lemmas 3.2 and 3.1. Based on these lemmas, an algorithm with asymptotic polynomial time complexity of $O(n^3)$ was proposed, allowing to obtain a numbering in any cograph G that induces the maximum possible quantity $\phi_{\pi}(G)$ of valid paths.

References

- Bloom, G. S. and Golomb, S. W. (1977). Applications of numbered undirected graphs. *Proceedings of the IEEE*, 65(4):562–570.
- Chung, F. R. K. (1981). Some problems and results in labelings of graphs. In *The theory and applications of graphs (Kalamazoo, Mich., 1980)*, p. 255–264. Wiley, New York.
- Dahlhaus, E., Gustedt, J., and McConnell, R. M. (2001). Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, 41(2):360–387. ISSN 0196-6774. URL <https://www.sciencedirect.com/science/article/pii/S019667740191185X>.
- Dias, E. S., Castonguay, D., Longo, H. J., and Jradi, W. A. R. (2013). Efficient enumeration of all chordless cycles in graphs. *CoRR*, abs/1309.1051. URL <http://arxiv.org/abs/1309.1051>.
- Foulds, L. and Longo, H. J. (2023). Valid path-based graph vertex numbering. URL <https://arxiv.org/abs/2306.02200>. arXiv eprint 2306.02200.
- Gallian, J. A. (2022). A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*. Twenty-first edition, #DS6.
- Judd, I. D. and Beer, R. (1995). Network addressing. U. S. Patent No. 5,465,251.
- Juvan, M. and Mohar, B. (1992). Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153–168.
- Leal, G. (2024). A study on the complexity of 2-path validity problems. Master’s thesis, Universidade Federal de Goiás.
- Rosa, A. (1967). On certain valuations of the vertices of a graph. In *Theory of Graphs (Internat. Symposium, Rome, 1966)*, p. 349–355. Gordon and Breach, N. Y. and Dunod Paris.
- Shieh, M. F., Yu, S. S., Yen, A., M, Y. S., Chang, C. Y., Xu, J. J., Wann, C. H., and Taiwan Semiconductor Manufacturing Co. (2011). Integrated circuit layout design. U. S. Patent No. 7,862,962.