Sequence graphs realizations and ambiguity in language models

Sammy Khalife, Yann Ponty, Laurent Bulteau

SEQUENCE GRAPHS REALIZATIONS AND AMBIGUITY IN LANGUAGE MODELS

SAMMY KHALIFE, YANN PONTY, AND YANN PONTY

Abstract. Several popular language models represent local contexts in an input text x as bags of words. Such representations are naturally encoded by a sequence graph whose vertices are the distinct words occurring in x, with edges representing the (ordered) co-occurrence of two words within a sliding window of size w. However, this compressed representation is not generally bijective, and may introduce some degree of ambiguity. Some sequence graphs may admit several realizations as a sequence, while others may not admit any realization. In this paper, we study the realizability and ambiguity of sequence graphs from a combinatorial and computational point of view. We consider the existence and enumeration of realizations of a sequence graph under multiple settings: window size w, presence/absence of graph orientation, and presence/absence of weights (multiplicities). When $w = 2$, we provide polynomial time algorithms for realizability and enumeration in all cases except the undirected/weighted setting, where we show the #P-hardness of enumeration. For $w \geq 3$, we prove hardness of all variants, even when w is considered as a constant, with the notable exception of the undirected/unweighted case for which we propose an XP algorithms for both (realizability and enumeration) problems, tight due to a corresponding W[1]-hardness result. We conclude with practical ILP and dynamic programming formulations for the problem. This work leaves open the membership to NP for both problems, a non-trivial question due to the existence of minimum realizations having exponential size on the instance encoding.

1. Introduction

The automated treatment of familiar objects, either natural or artificial, always relies on a translation into entities manageable by computer programs. A common challenge in data science is the choice of a vector or matrix representation, called *embeddings*, for a sequence of words from a vocabulary. Selecting and computing the right embedding therefore poses a central problem for the application of machine learning techniques. In particular, embeddings of words and textual documents representations are essential for several tasks in natural language processing, including document classification [15], role labelling [12], and named entity recognition [9]. Models based on pointwise mutual information, or *Graph-Of-Words* (GOW) [6, 13, 10] supplement the content of bag-of-words with statistics of co-occurrences within a *window* of fixed size w, thus mitigating the degree of ambiguity.

Johns Hopkins University, Department of Applied Mathematics and Statistics
CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
CNRS, Universite Gustave Eiffel, 77454 Marne-la-Vallee
E-mail addresses: khalife.sammmy@jhu.edu, yann.ponty@lix.polytechnique.fr, laurent.bulteau@u-pem.fr.

2020 *Mathematics Subject Classification.* 68T07, 68Q19, 05D10, 11J85.

Several models [8, 11, 1, 14] also use the same type of information and constitute strong baselines for natural language processing.

While these representations are more precise than the traditional bag-of-words, sometimes referred to as *Parikh vectors* in the literature, they still induce some level of ambiguity, *i.e.* a given graph can represent several sequences (see Figure 1 and 2 for illustrations). Our study aims at quantifying this level of ambiguity, seen as an algorithmic problem.

Definitions and problem statement

Let $x = x_1, x_2, ..., x_p$ be a finite sequence of discrete elements among a finite vocabulary X. Without loss of generality, we can suppose that $X = \{v_1, \cdots, v_n\}$. In the following, for any integer $p > 0$, let $[p] := \{1, ..., p\}$ and consider the following definition:

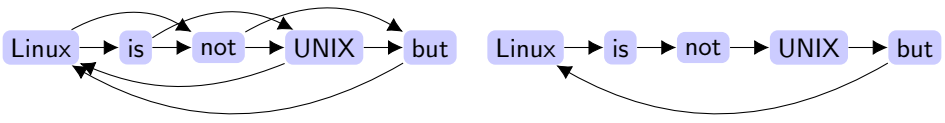
Definition 1. *$G = (V, E)$ is the sequence graph (or w-sequence graph) of the sequence x with window size $w \in \mathbb{N}^+$ ($w > 0$) if and only if $V = \{v \in X \,|\, \exists i \in [p],\ v = x_i\}$, and*

$$(1) \qquad (u, v) \in E \iff \exists (k, k') \in [p]^2 \quad 0 < |k - k'| \leq w - 1,\ u = x_k,\ v = x_{k'}$$

A sequence graph G is endowed with a weight matrix $\Pi(G) = (\pi_{ij})$ such that

$$(2) \qquad \pi_{ij} = \mathsf{Card}\ \{(k, k') \in [p]^2 \mid\ 0 < |k - k'| \leq w - 1,\ x_k = i \ \mathsf{and}\ x_{k'} = j\}$$

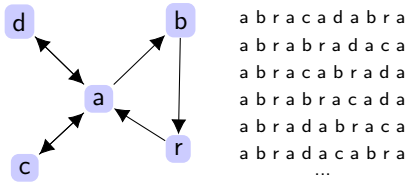
For digraphs, the absolute value in Statements (1) *and* (2) *is replaced with $k \leq k' \leq k + w - 1$. We say that x is a w-realization of G (or a realization if there is no ambiguity), if G is the graph of sequence x with window size w.*



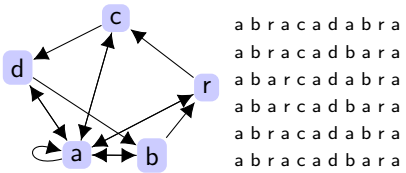
(A) Unambiguous graph ($w = 3$)

(B) Ambiguous graph ($w = 2$)

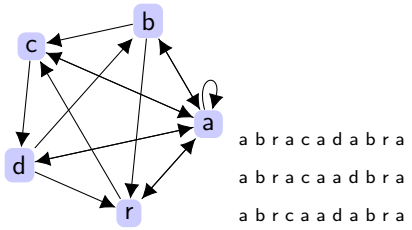
FIGURE 1. Sequence digraphs (or directed *graphs-of-words*) built for the sentence "Linux is not UNIX but Linux" using window sizes $w = 3$ (a) and $w = 2$ respectively (b). In the second case, the sequence graph is ambiguous, since any circular permutation of the words admits the same representation.



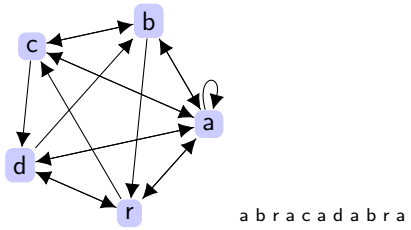
(A) $w = 2$, G has 30 realizations



(B) $w = 3$, G has 6 realizations



(C) $w = 4$, G has 3 realizations



(D) $w = 5$, G has one realization

FIGURE 2. Sequence digraphs (or directed *graphs-of-words*) built for the sentence "a b r a c a d a b r a" using window sizes 2 (a), 3 (b), 4 (c) and 5 (d).

Given w, the graph of a sequence x is unique. The natural integers π_{ij} represent the number of co-occurrences of i and j in all windows of size w. A linear time algorithm to construct a weighted sequence digraph is presented in Algorithm 1; the other cases (unweighted, undirected) are obtained similarly. In the unweighted case, the map thus defined from the sequence set $X^\star$ to the graph set $\mathcal{G}$ is referred to as $\phi_w \colon X^\star \to \mathcal{G}, x \mapsto G_w(x)$. Based on these definitions, we consider the following problems:

Problem 1 (Weighted-REALIZABLE (W-REALIZABLE))**.**
Input: *Graph G (directed or undirected), weight matrix Π, window size w*
Output: *True if (G, Π) is the w-sequence graph of some sequence x, False otherwise.*

Problem 2 (Unweighted-REALIZABLE (U-REALIZABLE))**.**
Input: *Graph G (directed or undirected), window size w*
Output: *True if G is the w-sequence graph of some sequence x, False otherwise.*

We denote *D*-REALIZABLE (resp. *G*-) the restricted version of REALIZABLE where the input graph G is directed (resp. undirected), and *W*-REALIZABLE (resp. *U*-) the restricted version of REALIZABLE where the input graph G is weighted (resp. unweighted), possibly in combination with the D- or G- variants. We write REALIZABLE$_w$ for the case where w is a fixed positive integer. We also consider the variants of W-REALIZABLE, denoted GW-REALIZABLE and DW-REALIZABLE where the input graph is restricted to be respectively undirected and directed. We define GU-REALIZABLE and DU-REALIZABLE similarly. Finally, we write (GW-, DW-, ...)REALIZABLE$_w$ for the case where w is a fixed positive integer.

Problem 3 (Unweighted-NUMREALIZATIONS (U-NUMREALIZATIONS))**.**
Input: *Graph G (directed or undirected), window size w*

Output: *The number of* ***realizations*** *of G,* i.e. *preimages of G through ϕ_w such that $|\{x \in X^\star \mid \phi_w(x) = G\}|$ is finite, or $+\infty$ otherwise.*

Problem 4 (Weighted-NumRealizations (W-NumRealizations))**.**
Input: *Graph G (directed or undirected), weight matrix Π, window size w*
Output: *The number of* ***realizations*** *of G in the weighted sense.*

Similarly, we use the same prefix for the directed or undirected versions of (D-, G-, i.e. DU- for directed and unweighted). We also denote NumRealizations$_w$ for the case where w is a fixed strictly positive integer. Note that NumRealizations generalizes the previous one, as Realizable can be solved by testing the nullity of the number of suitable realization computed by NumRealizations.

DW Directed weighted	**DU** Directed unweighted
GW Undirected weighted	**GU** Undirected unweighted

Algorithm 1 Construction of Π associated to a weighted sequence digraph

Parameters: Window size w
Input: Sequence x of length p, $p \geq w \geq 2$
Output: Weighted adjacency matrix Π

```
Initiate Π = (π_{i,j}) to an n × n matrix of zeros
for i = 1 → p − 1 do
    for j = i + 1 → min(i + w − 1, p) do
        π_{x_i,x_j} ← π_{x_i,x_j} + 1
    end for
end for
return Π
```

Related work

Sequence graphs encode the information of several co-occurences based models [1, 11]. To the best of our knowledge, the ambiguity and realizability questions addressed in this work were never addressed by prior work in computational linguistics. It may seem that the inverse problems we are considering in this work are similar to the *Universal Reconstruction of a String* [5], which consists in determining the set of strings of a fixed length having as many distinct letters as possible, satisfying substrings equations of the form: $s[q_1 \cdots q_p] = s[q'_1 \cdots q'_p], \cdots, s[r_1 \cdots r_m] = s[r'_1 \cdots r'_m]$ (here, $s[q_1 \dots q_p]$ refers to the substring $s_{q_1} \dots s_{q_p}$). The strictly increasing indices q_i's, q'_i's, $\cdots$, r_i's and r'_i's, as well as the length of s are given as input. The problem is to find a string s verifying these set of constraints, with a maximum number of distinct letters. We shall see that these problems are actually very different, and in particular, our complexity results imply the absence of reduction to the Universal Reconstruction of a String, which can be solved in linear time.

Furthermore, some similarities exist with another inverse problem studied in the Distance Geometry (DG) literature. The input of a DG instance consists of a set of pairwise distances between points, having unknown positions in a d-dimensional space. A DG problem then consists in determining a set of positions for the points (if they exist), satisfying the distance constraints. Since a position is fully characterized from $d+1$ neighbors, the problem can be solved by finding a sequential order in the

points, such that the assignment of a point is always by at least $d+1$ among its neighbors [7] (called *linear ordering*). Therefore, finding a linear ordering shares some level of similarity with our inverse problems since a realization for a window $w = d+2$ also represents a linear ordering of its nodes, in which $w-1 = d+1$ of the neighbors have lower value with respect to the order. However, linear ordering in DG to solve our problems is insufficient. First, each element of the sequence x is associated with a unique vertex. This is not the case we investigate here, since a symbol can be repeated several times, but only one vertex is created in the graph. This implies that the vertex associated to the i^{th} element ($i \geq w$) of x can have strictly less than $w-1$ distinct neighbors in its predecessors in x. Second, DG graphs are essentially undirected, and loops are not considered, since an element is at distance 0 from itself.

The remaining of the article is organized as follows. In Section 2 we present our main theoretical results. Full proofs are given in Sections 3 ($w = 2$) and 4 ($w \geq 3$). In Section 5, we propose an integer program and a dynamic programming algorithm to respectively recognize a sequence graph and count its realizations. Finally, in Section 6 we conclude with a short discussion and a first step towards an answer to the belonging of our problems to NP, by proving the existence of graphs whose minimal realizations have exponential size.

2. Theoretical results

In this section, we present our main theoretical results in Subsections 2.1 and 2.2. Full proofs are given in Sections 3 and 4 respectively.

2.1. A complete characterization of 2-sequence graphs.

A graph has a realization with $w = 2$ when there exists a path visiting every vertex and covering all of its edges (at least once for the unweighted case and exactly π_e for the edge e in the weighted case). This characterization enables relatively simple characterization and algorithmic treatment, leading to the results summarized in Table 1. The additional definitions are given below.

Table 1. Complexity for various instances of our problems ($w = 2$)

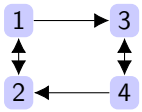
	NumRealizations$_2$		Realizable$_2$	
Data Instance	Complexity	#Sequences	Complexity	Characterization
GU	P	$\{0, +\infty\}$	P	G connected
GW	#P-hard	$\{0, 1\} \cup 2\mathbb{N}^*$	P	$\psi(G)$ (semi-)Eulerian
DU	P	$\{0, 1, +\infty\}$	P	G is a simple step
DW	P	$\mathbb{N}$ (BEST Theorem)	P	$\psi(G)$ (semi-)Eulerian

Definition 2. *Let $\psi(G)$ be the multigraph with the same vertices as $G = (V, E)$ and with π_{ij} edges between $(i, j) \in V^2$.*

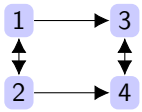
Definition 3. *A graph G (possibly oriented) is Eulerian if and only if there exists a cycle visiting every edge of G exactly once. We call such cycle a Eulerian path. G is semi-Eulerian if and only if there exists a path whose starting vertex is distinct from the last vertex, and visiting every edge of G exactly once. We call such path a semi-Eulerian path.*

Definition 4. *Let G be a digraph. $R(G)$ is the Directed Acyclic Graph (DAG) obtained by contracting its strongly connected components into a node; an edge between two new nodes is created if and only if there is an edge between two nodes of two strongly connected components of G. $R^+(G)$ is the weighted DAG obtained from $R(G)$, such that the weight of an edge is the number of distinct arcs from two strongly connected components in G.*

Definition 5. *Let G be a digraph. G is said to be a **simple step** graph if and only if $R^+(G)$ is a directed path and its weights are all equal to 1.*



(A) G is a simple step



(B) G is not a simple step

2.2. **Main complexity results for $w \geq 3$.** In this subsection we present the remaining complexity results, which are summarized in Theorem 1 and Table 2. We first show that GU-Realizable$_w \in P$, $\forall w \geq 3$. Besides, for GU, the number of realizations of a graph G is either 0 (not realizable), $+\infty$ (realizable and there exists a cycle in a component of H generating G), or 1 (realizable but no cycle in any component of H generating G). These three cases can be tested in polynomial time using our algorithm, showing that GU-NumRealizations$_w \in P$, $\forall w \geq 3$. All proofs of the following statements are given in Section 4.

Theorem 1. *For $w \geq 3$, all variations of* NumRealizations$_w$ *and* Realizable$_w$ *are NP-hard, except **GU**. Besides,* NumRealizations, Realizable *are para-NP-hard for all variations, except **GU**, in which case they are both W[1]-hard and XP.*

Table 2. Complexity for various instances of our problems ($w \geq 3$). We remind that a para-NP-hard problem does not admit any XP algorithm unless P=NP.

	Constant w, $w \geq 3$		Parameter w	
Variation	NumRealizations$_w$ Complexity	Realizable$_w$ Complexity	NumRealizations Complexity	Realizable Complexity
GU	P	P	W[1]-hard; XP	W[1]-hard; XP
GW	NP-hard	NP-hard	para-NP-hard	para-NP-hard
DU	NP-hard	NP-hard	para-NP-hard	para-NP-hard
DW	NP-hard	NP-hard	para-NP-hard	para-NP-hard

In the following, Clique is the problem which takes as input an undirected graph G and should return the maximal size of a clique in G.

Proposition 1. Clique *admits a polynomial time parameterized reduction to* GU-Realizable.

Corollary 1. GU-Realizable *is* W[1]*-hard for parameter w.*

3. The Special Case of 2-Sequence Graphs ($w = 2$)

In this section we present the proofs of the results gathered in Table 1. Apart from the **GU** variant, we use direct reductions to standard well-known problems in graph theory. The **DU** variant can be treated with a reduction to simple step graphs (cf. Definitions 4 and 5). The weighted cases (**GW** and **DW**) cases are treated with direct reductions to the problem of existence and counting Eulerian cycles or trails in a graph.

3.1. The unweighted variants: GU and DU.

The following three propositions follow immediately from the definitions:

Proposition 2. *If $G = (V, E)$ is unweighted and undirected, with $|V| > 1$, the following are equivalent:*
(i) G is connected
(ii) G has a 2-realization
(iii) G admits an infinite number of 2-realizations.
In these conditions, a 2-realization can start and end at any vertex.

The previous characterization is wrong for strongly connected digraphs. A counterexample is depicted in Fig. 3a. However, strong connectivity remains a sufficient condition:

Proposition 3. *Let $G = (V, E)$ a unweighted digraph. If G is strongly connected then G has a 2-realization. A 2-realization can start or end at any given vertex of G.*

Proposition 4. *Let $G = (V, E)$ an unweighted digraph. If G is Eulerian or semi-Eulerian, then G has a 2-realization.*

Again the converse of Prop. 4 does not hold as depicted in Fig. 3b. As a start, it is natural to consider directed acyclic graphs (DAGs):

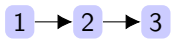
Proposition 5. *Let $G = (V, E)$ be a DAG. G is a 2-sequence graph if and only if it is a directed path, i.e G is a directed tree where each node has at most one child and at most one parent. In this case, G has a unique 2-realization.*

Proof. If G is directed path, since G is finite, it admits a source node. Therefore a 2-realization is obtained by simply going through all vertices from the source node. This is obviously the only one.

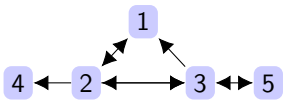
Conversely, let us suppose G is a DAG and a 2-sequence graph. If G is not a directed path, there are two cases: either there exists a vertex having two children, or two parents. Let s be a vertex having 2 distinct children c_1 and c_2. This is not possible since there cannot be a walk going through (s, c_1) and (s, c_2): G would have a cycle otherwise. Finally a vertex v cannot have two parents p_1 and p_2: if a 2-realization existed, it would have to go through (p_1, v) and (p_2, v), creating a cycle, hence the contradiction. □

Proposition 6. *Let $G = (V, E)$ a digraph. If G is a 2-sequence graph then $R(G)$ is a 2-sequence graph.*

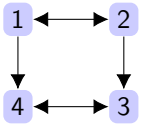
Proof. Let G be a 2-sequence graph, and let us suppose that $R(G)$ is not a 2-sequence graph. Since $R(G)$ is a (weakly) connected DAG, then using Prop. 5, it cannot be a directed path, so $R(G)$ has either a node having two children or two



(A) 1 2 3 is a 2-realization but is G is not strongly connected

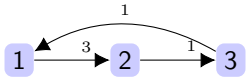


(B) 3 5 3 1 2 1 2 3 2 4 is a 2-realization but the graph is not semi-Eulerian



(C) G is not a 2-sequence graph...

(D) ... whereas $R(G)$ is.

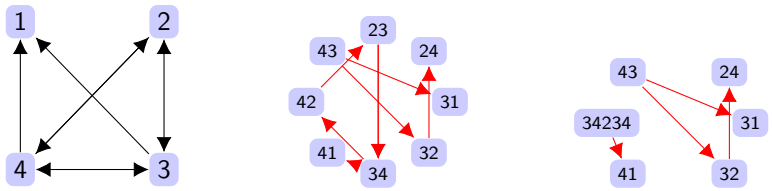


(E) G is strongly connected but is not a 2-sequence graph because of its weights

FIGURE 3. Counterexamples for $w = 2$

parents. Let S be a node of $R(G)$ having at least 2 distinct children C_1 and C_2. This means that there exist three distinct corresponding nodes in V, s, v_1 and v_2 such that $(s, v_1) \in E$ and $(s, v_2) \in E$. Since G is a 2-sequence graph, there exists a walk covering (s, v_1) and (s, v_2), such walk would make S, C_1 and C_2 the same node in $H(G)$, hence the contradiction. The case for which a vertex has two parents is dealt with similarly. □

The converse of Prop. 6 does not hold as depicted in Fig. 3c and 3d.



(A) G (B) H (C) $R(H)$

FIGURE 4. Procedure to find a 3-realization (**DU** variant). The walk 34234, 41 in $R(H)$ gives the 3-realization: 3 4 2 3 4 1

Theorem 2. *Let $G = (V, E)$ be an unweighted digraph. G is a 2-sequence graph if and only if it is a simple step.*

Proof. If G is a 2-sequence graph, $R(G)$ is a 2-sequence graph using Prop. 6. Also Prop. 5 implies that $R(G)$ and $R^+(G)$ are directed paths. Moreover, if $R^+(G)$ had a weight strictly greater that 1, then there would be more than one edge between two strongly connected components C_1 and C_2. All these edges go in the same direction otherwise $C_1 \cup C_2$ would form a strongly connected component. This is a contradiction since any 2-realization would have to go from C_1 to C_2 and then come back to C_1 (or conversely), which would would make $C_1 \cup C_2$ a strongly connected component.

Conversely, let us suppose $R^+(G)$ is a directed path and its weights are equal to one. By definition, there exists a list of sets of vertices $\mathcal{P} = (x_1, ..., x_p)$ such that:

(i) the entries of $\mathcal{P}$ form a partition of $V(G)$, i.e. $x_i \subset V(G)$ with $|x_i| \geq 1$, $\bigcup_{i\in\{1,\cdots,p\}} x_i = V(G)$ and for any $i \neq j$, $x_i \cap x_j = \emptyset$.

(ii) For any $i \in \{1, \cdots, p-1\}$, there exists a unique element of $x_i \times x_{i+1}$ which is an edge of G.

We construct a 2-realization y for G by means of the following procedure.

Base case: If x_1 is a singleton, i.e. $x_1 = \{v\}$ with $v \in V(G)$, we simply set $y \leftarrow v$. Otherwise, x_1 is a strongly connected component G and we add to y any of the 2-realizations of x_1.

For $i \in \{1, .., p-1\}$:

- If x_i and x_{i+1} are both singletons, we simply add the vertex of x_{i+1} to the sequence y.
- If x_i is a singleton and $|x_{i+1}| > 1$, then by assumption, there exists only one edge $e \in E(G)$ whose source is in x_i and target is in x_{i+1}, say $(v, w) \in E(G)$. Since the induced subgraph $G[x_{i+1}]$ is strongly connected, using Prop. 3, there exists a walk on $G[x_{i+1}]$ starting on w and going through every of its edges. We add the vertices of this walk to y.
- If $|x_i| > 1$ and x_{i+1} is a singleton, then there exists only one edge between a vertex of x_i and a vertex of x_{i+1} say $e = (v, w)$. By construction, all the edges of $G[x_i]$ have already been visited by a walk whose vertices have been added to y in the previous step. Suppose that the last vertex visited is z. Therefore, we add to y the vertices of walk from z to v that arrive strictly after z, and then w.
- If $|x_i| > 1$ and $|x_{i+1}| > 1$, then there exists only one edge between a vertex of x_i and a vertex of x_{i+1} say $e = (v, w)$. By construction, all the edges of $G[x_i]$

have already been added to y. Suppose at the previous step the last vertex added is $z \in x_i$. We first consider a walk starting at z and ending on w, and add all the vertices of this walk strictly after z. Then, consider a walk starting at w and which visits every edge of x_{i+1} (again the existence is such walk follows from Prop. 3). We add all vertices of this walk that arrive strictly after w.

End For

The process stops when $i = p - 1$, and all edges of G are covered by y. □

An immediate consequence of Theorem 2 is the existence of a polynomial time algorithm to decide if an unweighted digraph is a 2-sequence graph; because verifying that a digraph is a simply step is in P. Another consequence of Theorem 2 is the following:

Corollary 2. *Let G an unweighted digraph. The possible numbers of 2-realizations for G is exactly $\{0, 1, +\infty\}$. Moreover, G admits a unique 2-realization if and only if G is a directed path.*

Proof. Let G a 2-sequence graph and let us show that G has either a unique or an infinite number of 2-realizations. G verifies characterization of Theorem 2. If $R(G)$ has a vertex representing a strongly connected component of G (or a vertex with a self loop), then by adding an arbitrary number of cycles to y, the obtained walk is still a realization. Otherwise, if every vertex of $R(G)$ is in V without self-loops in E, then G is a DAG. Using Prop. 5, y is the unique 2-realization. □

3.2. **The weighted variants: GW and DW.** The weighted case cannot be treated similarly due to the weight constraints implying that a weighted graph has a finite number of realizations. A counterexample is depicted in Fig. 3e.

Theorem 3. *If G is a weighted graph (possibly directed), with $\Pi(G)$ a $n \times n$ matrix of natural integers, then: G is 2-realizable if and only if $\psi(G)$ is connected and (semi-)Eulerian.*

This theorem follows from the following stronger result, that also relates the number of 2-realizations to the number of (semi-)Eulerian paths of $\psi(G)$.

Lemma 1. *Let $G = (V, E)$ a weighted 2-sequence graph (possibly oriented). Let $\mathcal{E}$ be the set of (semi-)Eulerian paths of $\psi(G)$ and $\mathcal{S}$ be the set of 2-realizations of G. Then*

$$|\mathcal{E}| = |\mathcal{S}| \prod_{e \in E} \pi_e!$$

Proof. First note that (semi-)Eulerian paths of $\psi(G)$ (writing h for the number of edges in $\psi(G)$) can be characterized by a pair $(u_0u_1 \dots u_h, e_1 \dots e_h)$ where each u_i is a vertex of G, $e_1 \dots e_h$ is a permutation of the edges of $\psi(G)$, and $e_i = (u_{i-1}, u_i)$ (directed case) or $e_i = \{u_{i-1}, u_i\}$ (undirected case). Note that $u_0u_1 \dots u_h$ is a 2-realization of G, and that, conversely, a (semi-)Eulerian path can be obtained form any $u_0u_1 \dots u_h$ by taking e_i to be one copy of (u_{i-1}, u_i) or $e_i = \{u_{i-1}, u_i\}$ for each i (the path indeed goes through all π_{uv} copies of each edge between u and v in $\psi(G)$ by definition of weighted 2-realizations).

Consider the map:

$$\begin{aligned} f : \mathcal{E} &\longrightarrow \mathcal{S} \\ (u_0u_1 \dots u_h, e_1 \dots e_h) &\mapsto (u_0u_1 \dots u_h) \end{aligned} \tag{3}$$

We have already noted that f is surjective, however it is not necessarily injective (visiting multiple copies of the same edge in different orders give the same 2-realization but with different (semi-)Eulerian paths). An element $x \in \mathcal{E}$ can be thought of a list of edges of G, each appearing π_e times, since each edge $\psi(G)$ is obtained by copying π_e times every edge of G. Therefore this map is not injective, as soon as there is one $\pi_e > 0$, because one can permute the corresponding edges in the (semi-)Eulerian path, and the corresponding 2-sequence is the same.

We thus consider the following relation $\mathcal{R}$ on $\mathcal{E}$: For two (semi-)Eulerian paths P_1 and P_2, $P_1 \mathcal{R} P_2 \iff P_1$ can be obtained from P_2 by permuting edges of $\psi(G)$ that are copies of the same edge in G. $\mathcal{R}$ is an equivalence relation because it is symmetric, transitive and reflexive. Let $\mathcal{E}/\mathcal{R}$ be $\mathcal{E}$ quotiented by $\mathcal{R}$. We have $P_1 \mathcal{R} P_2 \iff f(P_1) = f(P_2)$ (equivalently, P_1 and P_2 yield the same sequence of vertices), so $|\mathcal{S}|$ is the number of equivalence classes of $\mathcal{R}$, or equivalently, $|\mathcal{E}/\mathcal{R}|$. Note that each equivalence class of $\mathcal{R}$ has cardinality $\prod_{e \in E} \pi_e!$ (number of permutations which are product of permutations with disjoint supports, where each support has size π_e). Therefore $|\mathcal{S}| = |\mathcal{E}/\mathcal{R}| = |\mathcal{E}|(\prod_{e \in E} \pi_e)^{-1}$. □

On the one hand, counting the number of (semi-)Eulerian paths in a undirected graph is a $\#P$-complete problem [2]. Since $G \mapsto \psi(G)$ is bijective, counting the number of 2-realizations is also $\#P$-complete. On the other hand, counting (semi-)Eulerian paths of a weighted digraph is in P, and can be derived using the following proposition:

Proposition 7. *Let $G = (V, E)$ be a weighted digraph, with $\Pi(G) \in \mathcal{M}_d(\mathbb{N})$. Then, if $deg(v)$ is the indegree of a vertex v, the number p_2 of 2-realizations is given by*

$$\text{-If } \psi(G) \text{ is Eulerian,} \qquad p_2 = \frac{t(\psi(G))}{\prod_{e \in E} \pi_e!} \prod_{v \in V} \big(\deg_{\psi(G)}(\psi(v)) - 1\big)! \tag{4}$$

where $t(G)$ is the number of spanning trees of a graph G. If L is the Laplacian matrix of G, then $t(G)$ can be expressed as

$$t(G) = \prod_{\substack{\lambda_i \in Sp(L) \\ \lambda_i \neq 0}} \lambda_i$$

- If $\psi(G)$ is semi-Eulerian, make it Eulerian by adding one arc (u, v) between the two vertices with unbalanced degrees (u is the one with the least outdegree, v has the least indegree). Then apply formula 4 to $\psi(G) + (u, v)$, and divide by the output by number of vertices $|V|$.

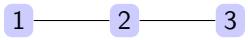
Proof. The case of $\psi(G)$ being Eulerian is a direct consequence of Lemma 1 BEST Theorem [3] and Matrix Tree Theorem [4].

When $\psi(G)$ is semi-Eulerian, this follows from the fact that $\psi(G)$ is semi-Eulerian if and only if $\psi(G) + (u, v)$ is Eulerian where: u is the the vertex whose outdegree is less than its indegree, and v is the vertex whose indegree is less than its outdegree. In that case, the number of semi-Eulerian paths of $\psi(G)$ is exactly the number of Eulerian paths of $\psi(G) + (u, v)$ divided by $|\psi(G)| = |V|$ (since for one semi-Eulerian path in $\psi(G)$ there are exactly $|V|$ Eulerian paths in $\psi(G) + (u, v)$). □

To relate formula 4 to the initial parameters of our problem, note that $deg_{\psi(G)}(\psi(v)) = \sum_{n \in V} \pi_{nv}$.

4. General Case with Arbitrary Window Size ($w \geq 3$)

The characterization of general sequence graphs, such as 3-sequence graphs is not the same for 2-sequence graphs, as shows the counterexample in Fig 5a: the depicted graph has no self-edge so there must be at least one clique of size 3. Similarly, Fig. 5b depicts a counterexample for directed graphs: G does not have loops, so if it had a 3-realization, such sequence must be of the form $\{1\,2\,3\,1..., 1\,3\,2\,1..., 2\,3\,1\,2..., 3\,2\,1\,3..., 2\,1\,3\,2...\}$ but then $(3, 1)$ would form an edge.



(A) G is connected but not a 3-sequence graph

(B) G is strongly connected but is not a 3-sequence graph

Figure 5. Counterexamples for $w = 3$

4.1. A polynomial time algorithm for GU-Realizable$_w$.

We first introduce a couple of definitions and notations for the gadgets used in our polynomial time algorithm to solve GU-Realizable$_w$.

Definition 6. *Let $G = (V, E)$ be an undirected graph. $H(G)$ is the directed graph defined as follows:*

$$V(H(G)) = \{(u, v) : u \in V(G), v \in V(G), \{u, v\} \in E(G)\}$$

and $e = (v_1, v_2)$, $f = (v_3, v_4)$ are adjacent in $H(G)$ if and only if:

$$v_2 = v_3 \text{ and } \{v_1, v_4\} \in E \tag{5}$$

An edge of $H(G)$ can be seen as an unique triplet v_1, v_2, v_4 where $\{v_1, v_2\}, \{v_1, v_4\} \in E$. When there is no ambiguity on the graph G considered, $H(G)$ will simply be referred to as H. We will also use $u_{i,j}$ as a shorthand for the pair (u_i, u_j) and $u_{1:k}$ for the k-tuple $(u_1, ..., u_k)$. For $k \geq 1$, we also define $H^{(k)} := (E^{(k)}, E^{(k+1)})$ to be the directed graph such that:

$$E^{(k)} = \{u_{1:(k+1)} \in V^{k+1} \mid u_{1:k} \in E^{(k-1)}, u_{2:(k+1)} \in E^{(k-1)} \wedge \{u_1, u_{k+1}\} \in E\} \tag{6}$$

In the description above, an element of $E^{(k)}$ (pair of elements of V^k verifying $u_{2:k} = v_{1:(k-1)}$ and $\{u_1, v_k\} \in E$) is represented as the $(k+1)$-tuple $u_{1:(k+1)}$. Alternatively:

$$H^{(0)} := G = (V, E) \qquad \forall k \in \mathbb{N}^*, \;\; H^{(k)} := \Phi(H^{(k-1)}) \tag{7}$$

where Φ transforms edges into vertices and creates edges following Eq. 6.

Remark 1. *By definition, a walk P in $H(G)$ is always of the form:*

$$P = (t_1, t_2), ..., (t_{p-1}, t_p) \text{ s.t } \forall i \in \{1, ..., p-1\}, \ (t_i, t_{i+1}) \in E \tag{8}$$

It is clear that if $H(G)$ is a 2-graph, then G is a 3-graph since there is a walk going through all edges of $H(G)$. The converse is not true in general as depicted in Fig. 4. In order to decide if $G = (V, E)$ has a realization in the general case, we recursively merge pairs of vertices, hence the definition. However, the number of vertices and edges of $H^{(k)}$ can increase exponentially with respect to k (the complete graph is an example). The next Proposition states a correspondence between realizations of the original graphs, and walks on $H^{(w-2)}$.

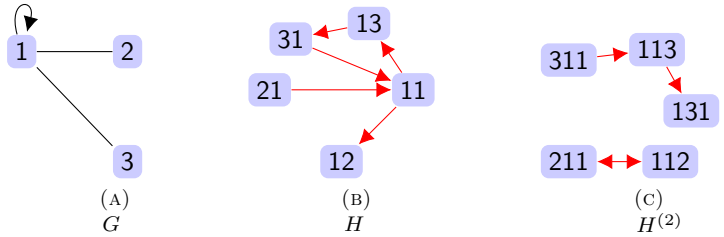


FIGURE 6. Example of construction of the gadgets H and $H^{(2)}$.

Definition 7. *Let u be a vertex of $H^{(k)}$ for $k \in \mathbb{N}$, $u = (u_1, ..., u_k, u_{k+1})$. The sequence $u_1, ..., u_{k+1}$ is the **authentic** sequence of u. We call an **authentic sequence** of a walk on $H^{(k)}$: $P = (x_1, ..., x_{k+1}), (x_2, ..., x_{k+2}), ..., (x_v, ..., x_{v+k})$ the sequence $x_1, x_2, ..., x_{v+k}$.*

Proposition 8. *Let $x = x_1, ..., x_p$ be a w-realization of a graph (or digraph) $G = (V, E)$. If $w \leq p$, then x is an authentic sequence of a walk of length $p - w + 1$ on $H^{(w-2)}$.*

Proof. If P is a walk on $H^{(w-2)}$, let $P[i]$ be the i-th element of P, $P[i] \in H^{(w-2)}$: $P[i] = (P[i]_1, ..., P[i]_{w-1})$. Let $x = x_1, ..., x_p$ be a w-realization of G.

We here suppose that $w \leq p$ (which we can always do), and show the following property by induction on k:

$$\forall k \in \{w-1, ..., p\}, \ \exists \text{ walk } P \text{ on } H^{(w-2)} \text{ such that :}$$

$$x_{1:k} = P[1]_1, P[2]_1, ..., P[k-(w-1)]_1, P[k+1-(w-1)]_{1:(w-1)}$$

• Base case: $k = w - 1$. $x_{1:w-1}$ is the authentic sequence of $P = P[1] = x_{1:w-1} \in H^{(w-2)}$.

• Induction step: let us suppose the property is verified for $k \in \{w-1, ..., p-1\}$, i.e there exists a walk P on $H^{(w-2)}$ such that:

$$x_{1:k} = P[1]_1, P[2]_1, ..., P[k-(w-1)]_1, P[k+1-(w-1)]_{1:(w-1)}$$

Since x is a w-realization, all the elements at distance at most w are edges of G:

$$\forall i \in \{k+1-(w-1), ..., k\}, \ \forall j \in \{i+1, ..., \min\{k+1, i+w-1\}\} : \ (x_i, x_j) \in E$$

This means in particular that $x_{k+1-(w-1)}, ..., x_{k+1} \in H^{(w-2)}$.

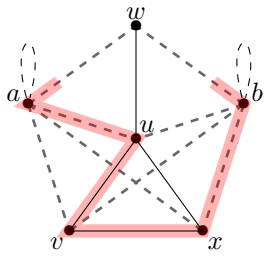


FIGURE 7. Illustration of the reduction for Proposition 1. The source graph G has vertices $\{u, v, w, x\}$. Vertices a and b and dashed arcs are added in the reduction. A realization follows a path visiting both vertices a and b: the first w vertices of the transition (highlighted in red) must form a clique in the graph.

Let $P[k+2-(w-1)]_{1:(w-1)} = x_{k+1-(w-1)}, ..., x_{k+1}$. From the induction assumption: $\forall i \in \{1, ..., k-(w-1)\},\ P[i]_1 = x_i$. This ensures that:

$$x_{1:(k+1)} = P[1]_1, P[2]_1, ..., P[k+1-(w-1)]_1, P[k+2-(w-1)]_{1:(w-1)}$$

which ends the induction and the proof. □

Proposition 9. *Let $w \in \mathbb{N}^+$ be any positive integer.* GU-REALIZABLE$_w$ *is in P.*

Proof. The case for $w = 1$ is trivial, and $w = 2$ has been treated. For $w \geq 3$, an algorithm is obtained by going through all the connected components of $H^{(w-2)}$. Let $C_1, ..., C_m$ the connected components of $H^{(w-2)}$. On the one hand, it is possible to compute them in polynomial time. On the other hand, one can construct walks covering all of their respective edges in polynomial time (for instance iteratively using shortest paths). Let $W_1, ..., W_m$ such walks and $X_1, ..., X_m$ their respective admissible sequences.

Using Prop. 8, G is a w-sequence graph if and only if there exists a walk $\tilde{W_{i_0}}$ on some C_{i_0} creating exactly the edges of G. However, W_{i_0} creates more edges than any walk on C_{i_0} by construction. In conclusion, the assertion: $\exists i \in \{1, ..., m\},\ \phi_w(X_i) = G$ is a characterization of G being a w-sequence. This assertion is decidable in polynomial time since for all i, $\phi_w(X_i)$ is computable in polynomial time (cf. Algorithm 1). □

Remark 2. *For digraphs, the analogue of the aforementioned procedure would consist in enumerating all paths in the DAG $R(H^{(w-2)})$. However, the number of those paths can be exponential, even if the initial graph is a sequence graph.*

Remark 3. *Proposition 9 provides a polynomial time algorithm for GU-*REALIZABLE$_w$*. If $x_1, ..., x_c$ are vertices of a strongly component C of $H^{(w-2)}$, one may wonder in which order should the attributes of the vertices be considered to form a new attribute x_C. This order is not important, as long as the walk visits every edge in the component. Moreover, it is possible to reconstruct all admissible sequences from walks on $R(H^{(w-2)})$.*

Proposition 1. CLIQUE *admits a polynomial time parameterized reduction to* GU-REALIZABLE.

Proof. Let $G = (V, E)$ be a simple graph. Let G' be a graph constructed from G adding two nodes a and b with loops, such that a and b are connected to each vertex of G. Let k be a strictly positive integer and $w = k + 1$. We will show that G has a k-clique if and only if G' is $((k + 1) = w)$-realizable.

First, let us suppose that G has a k-clique. Let C be an arbitrary sequence of the vertices of one of its k-cliques. Let $v_1, \ldots, v_{|V|}$ be the vertices of G and $\{u_1, u'_1\}, \ldots, \{u_{|E|}, u'_{|E|}\}$ be its edges. We write A (resp. B) for the string containing w successive copies of a (resp. b). Then, the following sequence is a w-realization of G':

$$A\ u_1\ u'_1\ A\ u_2\ u'_2\ A \ldots A\ u_{|E|}\ u'_{|E|}\ A \quad C \quad B\ v_1\ B\ v_2\ B\ \ldots B\ v_{|V|}$$

Now let us suppose that G' is w-realizable and let $x = x_1, \ldots, x_p$ be a w-realization of G'. Without loss of generality, we can suppose a appears before b in x. Let i_b be the index of the first appearance of b and let i_a be the largest index of the appearance of a before i_b. Then $i_b - i_a \geq w$, otherwise there would be an edge between a and b. Furthermore, since G is simple, there cannot be two repetitions of a vertex in the sequence $x_{i_a+1}, \ldots, x_{i_a+w-1}$. Due to the definition of a sequence graph, all vertices $\{x_{i_a+1}, \ldots, x_{i_a+w-1}\}$ are connected, forming a clique in G of size $w - 1 = k$, which ends the proof. □

4.2. **NP-Hardness Reductions.** We prove in this section our three NP-hardness for any constant window size (at least 3).

Proposition 10. *DU-Realizable$_w$, GW-Realizable$_w$, and DW-Realizable$_w$ are all NP-hard for any $w \geq 3$.*

We prove each case directly or indirectly by reduction from restricted versions of Hamiltonian Path. We first verify the NP-hardness of these variants (see lemma 2). We then focus on the unweighted case (see lemma 4), for which we introduce an intermediate variant with *optional* arcs. Finally for the weighted cases, we use the same reduction for both directed and undirected cases (simply ignoring arc orientations in the latter case, see lemma 5).

All our NP-hardness reductions are from Hamiltonian Path, where we require that the input graph contains up to two degree-one vertices. More formally, we reduce from the following intermediate problem:

Hamiltonian Variants The following is a folklore result, which we include here for completeness.

Lemma 2. *Hamiltonian Path is NP-hard even with either of the following two restrictions:*

- HP1 *the input graph has no self-loop, is directed and has a source vertex s (i.e. with in-degree 0)*
- HP2 *the input graph has no self-loop, is undirected and has two degree-1 vertices s and t.*

Proof. The first reduction is from Hamiltonian Cycle in directed graphs: pick any vertex v and duplicate it into v_1, v_2. Each arc (v, u) becomes (v_1, u) and each arc (u, v) becomes (u, v_2). Then any cycle in the orginal graph is equivalent to a path in the new graph from v_1 to v_2.

The second reduction is from Hamiltonian Cycle in undirected graphs: pick any vertex v and duplicate it into v_1, v_2. Eache edge $\{u, v\}$ becomes two edges $\{u, v_1\}$

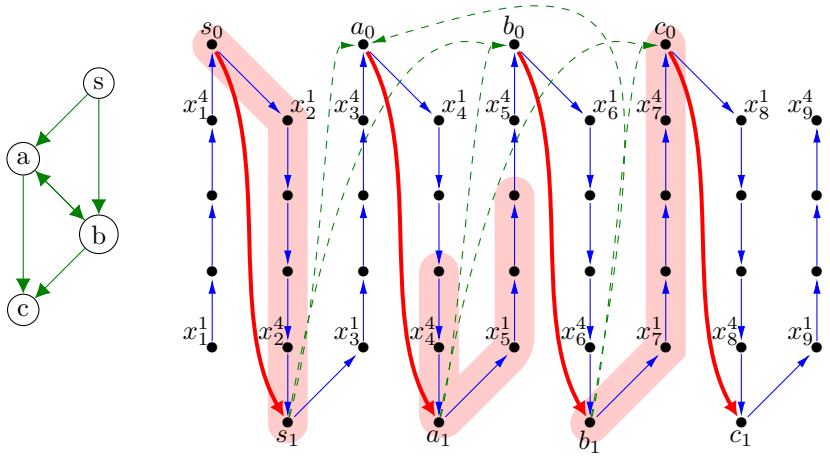


FIGURE 8. Left: an instance G of Hamiltonian Path with a source vertex s and solution (s, a, b, c). Right: the corresponding instance G' of OptionalRealizable$_6$. Heavy (red) arcs are compulsory, light (blue) arcs are a solution path in the graph, dashed (green) arcs are optional arcs issued from the intput graph. Other optional arcs are not depicted. Three size-6 windows are overlined: A window using v_0 and v_1 realizes the compulsory arc for vertex v, a window using u_1 and v_0 enforce that the arc (u, v) exists in G, other windows with $w-1$ separator vertices x_p^i help structure the whole sequence.

and $\{u, v_2\}$. Add pending vertices s and t connected to v_1 and v_2 respectively. Then any cycle in the orginal graph is equivalent to a path in the new graph with $\{s, v_1\}$ at one end and $\{t, v_2\}$ at the other. □

Reduction for DU-Realizable

In the directed and unweighted setting, we use the following intermediate generalization which allows *some* arcs to be ignored in the realization. For convenience in the final reduction, we further assume that the first $w-1$ elements of the sequence are given in input.

Problem 5. *OptionalRealizable$_w$*
Input: *directed unweighted graph $D = (V, A)$ without self-loops, a subset $A_c \subseteq A$ of* compulsory arcs*, a* starting sequence *$P = (s_1, \ldots, s_{w-1})$ of $w-1$ distinguished vertices of V.*
Question: *Is there a sequence S, starting with P, such that the graph of S with window size w contains only arcs in A and (at least) all arcs in A_c?*

Lemma 3. *For any fixed $w \geq 3$, OptionalRealizable$_w$ is NP-hard.*

Proof. By reduction from HAMILTONIAN PATH (see lemma 2, HP1). Given a directed graph $G = (V, A)$ with a source vertex s and no self-loop, build an instance of OptionalRealizable$_w$ with directed unweighted graph $G' = (V', A')$, compulsory arcs A'_c and starting sequence P as follows (see Figure 8 for an example).

We introduce vertices denoted v_0, v_1 for each vertex v of the original graph, as well as a grid of vertices x_p^i for each $1 \leq p \leq 2n+1, 1 \leq i \leq w-2$. The overall vertex set is thus

$$V' = \left(\bigcup_{v \in V} \{v_0, v_1\} \right) \cup \{x_p^i \mid 1 \leq p \leq 2n+1, 1 \leq i \leq w-2\}$$

The set of compulsory arcs is $A'_c = \{(v_0, v_1) \mid v \in V\}$. We further introduce the following optional arcs:

- arcs (u_1, v_0) for each (u, v) in A
- arcs (x^i_{2p-1}, v_0), (v_0, x^i_{2p}), (x^i_{2p}, v_1), (v_1, x^i_{2p+1}) for each $v \in V$, $1 \leq p \leq n$, $1 \leq i \leq w-2$.
- arcs (x_p^i, x_p^j) for $i < j$ and and $1 \leq p \leq 2n+1$; and (x_p^i, x_{p+1}^j) for $j \leq i$ and $1 \leq p \leq 2n$

The starting sequence is defined as $P = (x_1^1, \ldots, x_1^{w-2}, s_0)$. Note that the resulting graph has no self-loop.

Claim: (G', P, A'_c) is a yes-instance for $\mathsf{OptionalRealizable}_w \Leftrightarrow G$ admits a Hamiltonian path

$\Leftarrow$ Let v^p be the p-th vertex of V in the Hamiltonian path (and v_0^p, v_1^p be the corresponding vertices in G'). Without loss of generality, since s has degree 1, $v^1 = s$. For $1 \leq p \leq n$, let X^p be the sequence $x^1_{2p-1} \ldots x^{w-2}_{2p-1} v_0^p x^1_{2p} \ldots x^{w-2}_{2p} v_1^p$. Let $X^{n+1} = x^1_{2n} \ldots x^{w-2}_{2n}$, and S be the concatenation $X^1 \ldots X^{n+1}$. By construction S starts with P. Further, for each compulsory arc (v_0, v_1), if $v = v^p$, then compulsory arc (v_0, v_1) is realized in subsequence X^p. Finally, it can be checked that the graph of S contains only arcs of A'. Indeed, the sequence uses the following arcs: (v_0, v_1) for each v (which are compulsory arcs), arcs (v_1^p, v_0^{p+1}) for each arc (v^p, v^{p+1}) of the hamiltonian path, so $(v^p, v^{p+1}) \in A$ and $(v_1^p, v_0^{p+1}) \in A'$, arcs with an endpoint v_i and an endpoint x_p^j (which satisfy the parity conditions so they belong to A'), and finally arcs of the form (x_i^p, x_j^q), either with $q = p$ (in which case $i < j$) or with $q = p+1$ (in which case by the window size we have $j \leq i$): both kinds are also in A'.

$\Rightarrow$ Consider a sequence S, an occurrence of x_p^i in S for some $1 \leq i \leq w-2$, $1 \leq p \leq 2n$ (note that $p \neq n+1$), and let S' be the subsequence of S containing the $w-1$ characters following x_p^i. Let $T = x_p^{i+1} \ldots x_p^{w-2}$ and $U = x_{p+1}^1 \ldots x_{p+1}^i$ (note that T is possibly empty). T and U are seen both as strings and as sets of vertices. The out-neighborhood of x_p^i contains all vertices of $T \cup U$, as well as all vertices v_q for $v \in V$, where $q = 0$ if p is odd and $q = 1$ if p is even. Since there are $k-2$ vertices in $T \cup U$, and no vertex has a self-loop, then by the pigeon-hole principle string S' must contain at least one vertex v_q, $v \in V$. Since there are no arc (v_q, v'_q) for $v, v' \in V$, S' contains exactly one vertex v_q, thus it also contains all vertices of $T \cup U$. Based on the direction of the arcs in $T \cup U \cup \{v_q\}$, it follows that $S' = T \cdot v_q \cdot U$.

Let X_p be the string $x_p^1 \ldots x_p^{w-2}$. From the arguments above, and the fact that S starts with X_1 (since P uses vertices of X_1), there exist indices $i_1, j_1, \ldots, i_n, j_n$ such that

$$S = X_1 v^0_{i_1} X_2 v^1_{j_1} X_3 v^0_{i_2} X_4 v^1_{j_3} X_5 \ldots X_{2n+1}$$

From the window size w, there must exist an arc $(v^0_{i_p}, v^1_{j_p})$ for each p, so by construction $i_p = j_p$. Furthermore, these arcs are compulsory for each vertex v^0, so

$(i_1, \ldots, i_n)$ is a permutation of $\{1, \ldots, n\}$. Finally, there also exist an arc $(v^1_{j_p}, v^0_{i_{p+1}})$ in G', so there exists an arc $(v_{i_p}, v_{i_{p+1}})$ in G. Thus, $(v_{i_1}, \ldots, v_{i_n})$ is a Hamiltonian path in G. □

We can now prove that DU-Realizable$_w$ is NP-hard by reduction from OptionalRealizable$_w$.

Lemma 4. *For any fixed $w \geq 3$, DU-Realizable$_w$ is NP-hard.*

Proof. Assume that we are given a directed unweighted graph $G = (V, A)$, a subset $A_c \subseteq A$ of compulsory arcs (let $A_o = A \setminus A_c$ be the set of optional arcs), and a starting sequence $P = (s_1 \ldots s_{w-1})$ of vertices of V. The following reduction is illustrated in fig. 9.

Let $m = |A_o|$, write $A_o = \{(u_1, v_1), \ldots, (u_m, v_m)\}$. Create G' by adding $w(m+1) + m$ *separator* vertices: $w(m+1)$ vertices y^i_p with $1 \leq p \leq m+1$ and $1 \leq i \leq w$, and m vertices z_p for $1 \leq p \leq m$. Build the strings

$$Z = \left(\prod_{p=1}^{m} (y^1_p \ldots y^w_p u_p z_p v_p) \right) y^1_{m+1} \ldots y^w_{m+1}$$

$$Z' = Z s_1 \ldots s_{w-1}$$

The arc set can be concisely defined as follows : take the set A and insert all arcs realized by Z' involving at least one separator vertex to G'. In details, the additional arcs are the following (where indices i, j, p necessarily satisfy $1 \leq i \leq w$, $1 \leq j \leq w$ and $1 \leq p \leq m$):

- (y^i_p, y^j_p) for $i < j$ and $y^i_p, y^j_{p+1})$ for $j \leq i - 4$,
- (y^i_{m+1}, y^j_{m+1}) for $i < j$ and (y^i_{m+1}, s_j) for $j < i$,
- (y^i_p, u_p) for $2 \leq i$ and (u_p, y^j_{p+1}) for $i \leq w - 3$,
- (y^i_p, z_p) for $3 \leq i$ and (z_p, y^i_{p+1}) for $i \leq w - 2$,
- (y^i_p, v_p) for $4 \leq i$ and (v_p, y^j_{p+1}) for $i \leq w - 1$,
- (u_p, z_p) and (z_p, v_p).

Claim: G has a realization with optional arcs $\Leftrightarrow$ G' has a realization

$\Rightarrow$ Build a realization for G' by concatenating Z with the realization for G starting with $s_1 \ldots s_{w-1}$. All optional arcs of G' are realized in Z, all compulsory arcs of G' are realized in the suffix (the realization of G'), and all arcs involving a separator are realized in Z'. No forbidden arc is realized.

$\Leftarrow$ Let S be a realization of G'. We prove by induction on q, for $1 \leq q \leq |Z|$, that (i) S and Z' have the same prefix of length-$(q + w - 1)$ and (ii) any separator in $Z[1, \ldots, q]$ may only appear in $S[1, \ldots, q]$.

For $q = 1$, this is obtained by the fact that $Z[1] = y^1_1$ has in-degree 0 in G' (so S starts with y^1_1) and its out-neighborhood forms a size-$(w-1)$ tournament corresponding to $Z[2...w]$, so the length-w prefix of S is $Z[1...w]$. Consider now $1 < q \leq |Z|$. By induction S and Z' have the same prefix of length-$(q + w - 2)$, and separators up to position $q - 1$ in Z do not have any other occurrence in S. Let $q' = q$ if $S[q]$ is a separator (case A), and $q' = q + 1$ otherwise (case B). In both cases, $S[q']$ is a separator, its in-neighborhood contains at least one separator $Z[q-1]$ or $Z[q-2]$, so in particular vertex $S[q']$ may not have any other occurrence in the sequence (otherwise $Z[q-1]$ and/or $Z[q-2]$ would also have two occurrences). Furthermore, the out-neighborhood of $S[q']$ is $N = \{Z'[q'+1], \ldots, Z'[q'+w-1]\}$ without self-loops, so $S[q'+1, \ldots, q'+w-1]$ is a permutation of N. In case A, $w-2$

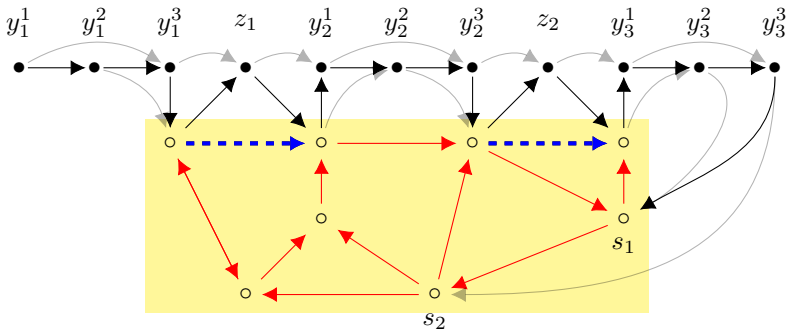


FIGURE 9. Reduction from OptionalRealizable$_w$ to DU-Realizable$_w$ with $w = 3$. The input instance is the highlighted graph with white vertices (including two optional dashed blue arcs), as well as the starting sequence (s_1, s_2). The reduction adds the black vertices y_p^i and z_p, and the corresponding black and grey arcs. Any solution is thus forced to first realize all optional arcs, and then realize the rest of the graph starting with s_1, s_2, and including all compulsory arcs and, as needed, some of the optional arcs again.

vertices of N are already accounted for (by induction) in $S[q'+1, \dots, q'+w-2]$, so the remaining vertex $Z'[q'+w-1]$ must be in position $q'+w-1$ in S. In case B, elements of N are all in Z, so they form a tournament and, again, the next $w-1$ positions in S and Z' must be equal.

Overall, we have $S = ZS'$ with the following properties: the length-$(w-1)$ prefix of S' is the starting sequence P, and no separator appears in S'. Thus S' realizes only arcs from G. Moreover no compulsory arc of G is realized in Z, nor with one vertex in Z and one in S' (since such arcs start with a separator), so all compulsory arcs are realized in S'. Overall, G is a yes-instance of OptionalRealizable$_w$ with sequence S'.

□

Now, let us prove that GW-Realizable$_w$ and DW-Realizable$_w$ are NP-hard for all $w \geq 3$, by reduction from Hamiltonian Path (see lemma 2, HP2). We focus on the directed case first, the undirected case will simply use the underlying graph introduced in this reduction.

Lemma 5. *For any fixed $w \geq 3$, DW-Realizable$_w$ and GW-Realizable$_w$ are NP-hard.*

Proof. **Reduction for DW-Realizable**

Given $G = (V, E)$ with degree-1 vertices s and t, write d_u for the degree of each vertex $u \in V$, and $k = w - 2$ (note that $k \geq 1$ since we chose $w \geq 3$). We write $\delta_s^{in} = d_s$ and $\delta_u^{in} = d_u - 1$ for $u \in V \setminus \{s\}$; and $\delta_t^{out} = d_t$ and $\delta_u^{out} = d_u - 1$ for $u \in V \setminus \{t\}$ (δ_u^{in} and δ_u^{out} can be seen as the remaining in- and out-degree in the oriented graph where edges are replaced by double arcs after removing an hamiltonian $s-t$ path). We write $\delta_u = \delta_u^{in} + \delta_u^{out}$. Build a directed weighted graph $G' = (V', A)$ as follows. For each $u \in V$, add u and a new vertex denoted u' to V'. Create additional dummy vertices s_0, s_0', a and b. The overall vertex set is thus

$V' := \{a, b, s_0, s'_0\} \cup \bigcup_{u \in V} \{u, u'\}$. The arcs of A are given in Figure 10, as the union of the start gadget, the queue gadget, and the vertex and edge gadgets respectively for each vertex and edge of G.

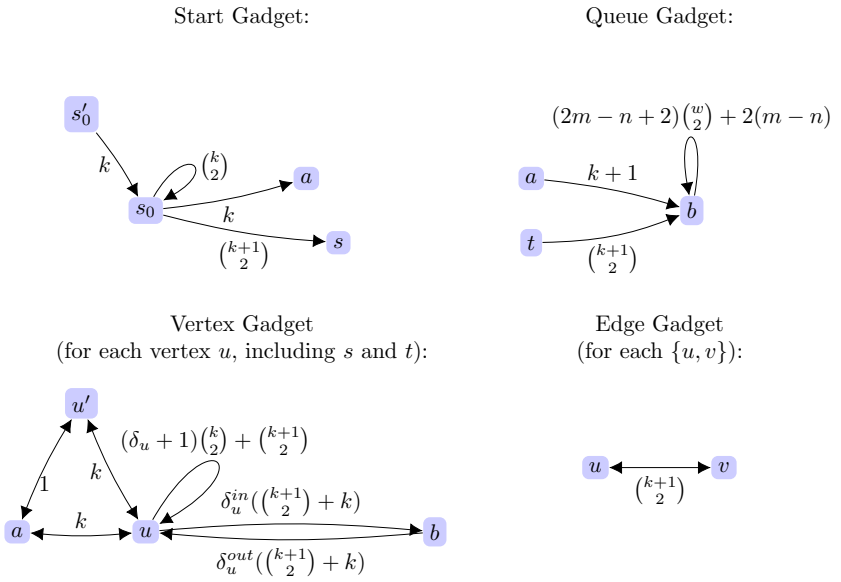


FIGURE 10. Subgraphs used in the reduction from Hamiltonian Path to DW-Realizable$_3$. Weights on double arcs apply to both directions. Note that arcs (t, b) appear in two different gadgets, so their weights should be summed

Reduction for GW-Realizable

Build the directed graph G' as above, and let G'_u be the undirected version of G': remove arc orientations, for $u \neq v$ the weight of $\{u, v\}$ is the sum of the weight of (u, v) and (v, u) in G' (the weight of loops is unchanged).

We prove the following three claims:
(i) G Hamiltonian $\Rightarrow$ G' has a realization
(ii) G' has a realization $\Rightarrow$ G'_u has a realization
(iii) G'_u has a realization $\Rightarrow$ G is Hamiltonian

Proof of Claim (i). Assume that G has a Hamiltonian path and denote its vertices as $u_1, u_2, \dots u_n$ according to their position along the path (wlog., $u_1 = s$ and $u_n = t$). Let $(v_1, w_1), \dots (v_{m'}, w_{m'})$ be the pairs of connected vertices in G that are not consecutive vertices of the hamiltonian path (formally, it corresponds to the set $\bigcup_{\{u,v\} \in E} \{(u, v), (v, u)\} \setminus \{(u_i, u_{i+1}) \mid 1 \leq i < n\}$). Note that there are $m' = 2m - (n-1)$ such pairs. We now show that the sequence S defined as follows is a realization of G.

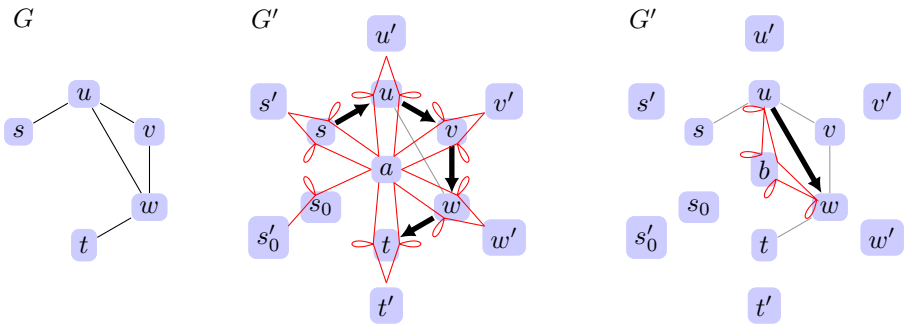


FIGURE 11. Reduction from Hamiltonian Path to DW-Realizable. Left: the input graph G with degree-one vertices s and t (and hamiltonian Path (s, u, v, w, t)). Each edge becomes two arcs (in each direction) in the edge gadgets of the resulting graph G'. Center: the first part of the path realizing most edges of G' (in particular those involving vertex a), following the hamiltonian path. In particular, bold arcs from the input graph are realized. Right: the remaining arcs (such as (u, w), but also (w, u), (u, s), (v, u), etc.) are realized using a succession of round-trips with vertex b. Self-loops represent iterations of $k = w - 2$ or w occurrences.

$$
\begin{aligned}
S &:= S_{init} S_{path} S_{queue} \quad \text{with} \\
S_{init} &:= s'_0\, s_0^k \\
S_{path} &:= a\, s^k\, s'\, s^k\, a\, u_2^k\, u'_2\, u_2^k\, a \ldots a\, u_{n-1}^k\, u'_{n-1}\, u_{n-1}^k\, a\, t^k\, t'\, t^k\, a \\
S_{queue} &:= b^w\, v_1^k\, b\, w_1^k\, b^w\, v_2^k\, b\, w_2^k \ldots b^w\, v_{m-n}^k\, b\, w_{m-n}^k\, b^w
\end{aligned}
$$

Note that a sequence of the form $x^k\, a\, y^k$ yields $\binom{k}{2}$ loops for x, $\binom{k}{2}$ loops for y, as well as $\binom{k}{2}$ arcs (x, y) (indeed, there are $1 + 2 + \ldots + w - 2 = \binom{k}{2}$ such arcs). A sequence of the form $b\, x^k\, b^w$ yields in particular an arc (b, x) of weight k and arc (x, b) of weight $\binom{w}{2}$.

We verify for each gadget that all arcs are indeed realized with the correct weight. Indeed, the start gadget corresponds exactly to arcs in S_{init} or overlapping S_{init} and S_{path}. Regarding the vertex gadget for $u \in V$, S_{path} realizes all arcs involving two distinct vertices among a, u, u'. S_{path} also yields $\binom{k}{2} + \binom{k+1}{2}$ self-loops for u, and S_{queue} yields the remaining $\delta_u \binom{k}{2}$ self-loops (since each vertex appears δ_u times there). S_{queue} also realizes all arcs between u and b. For an edge gadget $\{u, v\}$ if (u, v) (resp. (v, u)) is part of the hamiltonian path, then the arc is realized in S_{path}, otherwise it is realized in S_{queue}. Finally, the arcs in the queue gadget are realized either in S_{queue}, either as overlapping arcs between S_{path} and S_{queue}.

□

Proof of Claim (ii). Clear, any realization for G' is a realization for G'_u. □

Proof of Claim (iii). Pick a realization S of G'_u. Define the weight of a vertex in G_u as the sum of the weights of its incident edges (counting loops twice). From the construction, we obtain the following weights for a selection of vertices:

- s'_0 has weight $w-1$
- u' has weight $2(w-1)$ for $u \in V$
- a has weight $2(n+1)(w-1)$

From the weight of s'_0, it follows that this vertex must be an endpoint of S (wlog, S starts with s'_0). It follows that for any other vertex v with weight $2i(w-1)$, v must have exactly i occurrences in S (in general it can be either i or $i+1$, but if v has $i+1$ occurrences it must be both the first and last character of S, i.e. $v = s'_0$: a contradiction). Thus each u' occurs once and a occurs $n+1$ times in S.

Each u' occurs once, so order vertices of V according to their occurrence in S (i.e. $V = \{u_1, \ldots, u_n\}$ with u'_1 appearing before u'_2, etc.). For each i, the neighborhood of u'_i in S contains a twice, one a on each side (since there is no (a,a) loop). Other neighbors of u'_i may only be occurrences of u_i, so each u'_i belongs to a factor, denoted X_i, of the form $au_i^*u'_iu_i^*a$. Two consecutive factors X_i, X_{i+1} may overlap by at most one character (a), and if they do, then there exists an arc (u_i, u_{i+1}) in A , hence an edge $\{u_i, u_{i+1}\}$ (since $w \geq 3$) in E. There are n such factors X_{u_i}, and only $n+1$ occurrences of a, so all as except extreme ones belong to the overlap of two consecutive X_is, and there exists an edge $\{u_i, u_{i+1}\}$ for each i. Thus $(u_1, \ldots, u_n)$ is a Hamiltonian path of G. □

All together, claims (i), (ii) and (iii) show the correctness of the reductions for both GW-Realizable and DW-Realizable since they yield :
G is Hamiltonian $\Leftrightarrow$ G' has a realization
G is Hamiltonian $\Leftrightarrow$ G'_u has a realization □

5. Effective general algorithms

5.1. Realizable$_w$ **Linear integer programming formulation.** Let $G = (V, E)$ be a graph with integer weights $\pi_{e \in E}$. In this model, we represent a sequence x over the alphabet $\{1, ...n\}$, as a $(0-1)$ matrix $X \in \mathbb{M}_{n,p}(\{0,1\})$ encoding the sequence x:

$$X_{i,j} = \begin{cases} 1 & \text{if } x_j = i \\ 0 & \text{otherwise} \end{cases}$$

We represent the set of sequences over the alphabet $\{1, ...n\}$ by the $(0-1)$ matrices such that $\forall j \in \{1, ..., p\}, \sum_{i=1}^{n} X_{i,j} = 1$.

Given a window size w, a unit of $\pi_{e=(v_1,v_2)}$ corresponds to the appearance of two elements v_1, v_2 at a distance $i \in \{1, ..., w-1\}$ in the sequence. Now, let us consider a fixed distance i, and a starting index $j \in \{1, ..., p-i\}$, we use a intermediary slack variable $y_j^e(i) \in \{0,1\}$ to model the presence of such appearance using the constraint:

$$X_{v_1,j} X_{v_2,j+i} = y_j^e(i) \tag{9}$$

Then, the Boolean variable $y_j^e(i)$ is equal to 1 when v_1 is located at position j and v_2 at position $j+i$. We linearise Eq. 9 as:

$$\begin{aligned} -X_{v_1,j} \quad\quad\quad\quad + y_j^e(i) &\leq 0 \\ - X_{v_2,j+i} + y_j^e(i) &\leq 0 \\ X_{v_1,1} + X_{v_2,j+i} - y_j^e(i) &\leq 1 \end{aligned} \tag{10}$$

Each slack variable $y_k^e(i)$ is attributed to an edge e, a relative distance $i \in \{1, ..., w-1\}$ and a starting position $k \in \{1, ..., p-i\}$. Given our constraint formulation, every slack variable is attributed 3 constraints. For a digraph, the number of possible pair positions for a unit of $\pi_{e=(v_1,v_2)}$ is given by:

$$C = \sum_{i=1}^{w-1}(p-i) = p(w-1) - \frac{w(w-1)}{2} = (w-1)(p-\frac{w}{2})$$

Therefore, in our model, C corresponds to the number of slack variables attributed to constraints for an edge of the graph.

On the contrary, the absence of an edge $e = (v_1, v_2)$, corresponding to $\pi_e = 0$, can be modeled for a distance $i \in \{1, ..., w-1\}$ and a starting position $j \in \{1, ..., p-i\}$ as:

$$X_{v1,j} + X_{v_2,j+i} \leq 1$$

Then, $\textsc{Realizable}_w$ can be formulated as a linear integer program:

$$\min_{X \in \{0,1\}^{p \times n}, y \in \{0,1\}^{|E| \times C}} \sum_{e \in E} \sum_{i \in \{1,...,w-1\}} y_1^e(i) + ... + y_{p-i}^e(i)$$

under the constraints

$$\forall j \in \{1, ..., p\} \qquad \sum_{i=1}^{n} X_{i,j} = 1$$

$$\begin{array}{l} \forall e = (v_1, v_2) \in E \\ \forall e^{'} = (v_1^{'}, v_2^{'}) \notin E \\ \forall i \in \{1, ..., w-1\} \end{array} \left\{ \begin{array}{llll} -X_{v_1,1} & + y_1^e(i) \leq 0 & & \\ - X_{v_2,1+i} & + y_1^e(i) \leq 0 & & \\ X_{v_1,1} + X_{v_2,1+i} & - y_1^e(i) \leq 1 & X_{v_1',1} + X_{v_2',1+i} & \leq 1 \\ \vdots & & \vdots & \\ -X_{v_1,p-i} & + y_{p-i}^e(i) \leq 0 & X_{v_1',p-i} + X_{v_2',p} & \leq 1 \\ - X_{v_2,p} & + y_{p-i}^e(i) \leq 0 & & \\ X_{v_1,p-i} + X_{v_2,p} & - y_{p-i}^e(i) \leq 1 & & \end{array} \right.$$

$$\text{and } \forall e \in E \qquad \sum_{i \in \{1,...,w-1\}} y_1^e(i) + ... + y_{p-i}^e(i) \geq \pi_e$$

If the objective function reaches $\sum_{e \in E} \pi_e$ at its minimum then the output of $\textsc{Realizable}_w(G, \Pi)$ is True, and False otherwise.

5.2. NumRealizations$_w$ **Dynamic programming formulation.** We did not present a way to count realizations in the general case. We present in this subsection a method based on dynamic programming valid for all cases.

The recursion proceeds by extending a partial sequence, initially set to be empty, keeping track of for represented edges along the way. Namely, consider $N_w[\Pi, p, \mathbf{u}]$ to be the number of w-realizations of length p for the graph $G = (V, E)$, respecting a weight matrix $\Pi = (\pi_{ij})_{i,j\in V^2}$, preceded by a sequence of nodes $\mathbf{u} := (u_1, \ldots, u_{|\mathbf{u}|}) \in V^\star$. It can be shown that, for all $\forall p \geq 1$, $\Pi \in \mathbb{N}^{|V^2|}$ and $\mathbf{u} \in V^{\leq w}$, $N_w[\Pi, p, \mathbf{u}]$ obeys the following formula, using the notations of Section 4:

$$N_w\left[\Pi, p, \mathbf{u}\right] = \sum_{v\in V} \begin{cases} N_w\left[\Pi'_{(\mathbf{u},v)}, p-1, (u_1, ..., u_{|u|}, v)\right] & \text{if } |\mathbf{u}| < w-1 \\ N_w\left[\Pi'_{(\mathbf{u},v)}, p-1, (u_2, ..., u_{w-1}, v)\right] & \text{if } |\mathbf{u}| = w-1 \end{cases} \tag{11}$$

with $\Pi'_{(\mathbf{u},v)} := (\pi_{ij} - |\{k \in [1, |\mathbf{u}|] \mid (u_k, v) = (i, j)\}|)_{(i,j)\in V^2}$. The base case of this recurrence corresponds to $p = 0$, and is defined as

$$\forall\, \Pi,\ N_w[\Pi, 0, \mathbf{u}] = \begin{cases} 1 & \text{if } \Pi = (0)_{(i,j)\in V^2} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The total number of realizations is then found in $N_w[\Pi, p, \varepsilon]$, *i.e.* setting $\mathbf{u}$ to the empty prefix ε, allowing the sequence to start from any node.

The recurrence can be computed in $\mathcal{O}(|V|^w \times \prod_{i,j\in V^2}(\pi_{i,j} + 1))$ time using memoization, for p the sequence length. The complexity can be refined by noting that:

$$\sum_{i,j\in V^2} \pi_{i,j} \leq w \times p$$

To investigate the worst case scenario, we can consider the optimisation problem:

$$\max_\Pi \prod_{i,j\in V^2}(\pi_{i,j} + 1) \text{ such that } \sum_{i,j} \pi_{i,j} = w\, p. \tag{13}$$

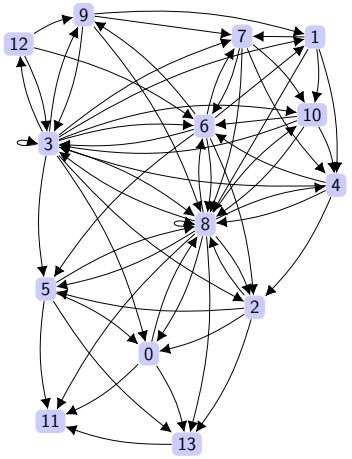
This problem is equivalent to maximise a product under a budget constraint. When $n^2 \geq w \times p$, which is the case in practice, the maximum is reached for a Boolean matrix $\Pi = (\pi_{i,j}) \in \{0, 1\}^{|V|^2}$, verifying the constraint. This property can be deduced from the inequality:

$$\begin{aligned} 1 \leq a < b - 1 &\implies \log a + \log b < \log(a + 1) + \log(b - 1) \\ &\implies ab < (a + 1)(b - 1) \end{aligned}$$

It follows that, in the worst-case scenario, $\prod_{i,j\in V^2}(\pi_{i,j} + 1) \in \mathcal{O}(2^{w\,p})$, giving an overall complexity of $\mathcal{O}(|V|^w 2^{w\,p})$. Thus, despite the apparently high complexity of our algorithm, it is still possible to compute $N_w[\Pi, p, u_{1:w}]$ for "reasonable" values of p and w. Indeed, succinct experiments showed that the table could be computed in less than a minute for values up to $|V| = 20$, $p = 100$ and $w = 3$. See Figure 12 for an instance and the resulting sequences obtained by our algorithm.

6. Discussion and open problems

In this study, we presented a new series of inverse problems related to the ambiguity of popular representations in text mining and natural language processing. We characterized their complexity class, except the belonging in NP for $w \geq 3$. Given a sequence, computing its graph representation can be done in $O(d^2 + p)$, if



(A) Sequence graph

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&1&0&0&1&0&1\\
0&0&0&0&1&0&0&1&1&0&1&0&0&0\\
1&0&0&0&0&1&0&0&1&0&0&0&0&1\\
1&2&1&2&0&1&2&1&2&2&1&0&1&0\\
0&0&1&1&0&0&1&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&1&0&1\\
0&1&1&2&0&1&0&1&1&1&0&0&0&0\\
0&0&0&0&1&0&1&0&1&0&1&0&0&0\\
1&0&1&1&1&1&1&0&1&0&1&1&0&1\\
0&1&0&1&0&0&0&1&1&0&0&0&0&0\\
0&0&0&1&1&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&2&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0
\end{pmatrix}$$

(B) Weights matrix Π

3 12 3 6 9 3 1 7 8 10 4 6 8 3 2 5 0 8 13 11

3 12 3 6 9 3 1 7 8 10 4 6 3 8 2 5 0 8 13 11

(C) Different realizations

FIGURE 12. Example of realizations in the **DW** variant, as obtained using our dynamic programming algorithm: (a) a 5-sequence graph on $|V| = 14$ (vertices are labelled with integers from 0 to 13). (b) the corresponding weight matrix Π of size 14×14. (c) two possible realizations of length $p = 20$.

p is the length of the sequence and d the size of the vocabulary. However, this does not prove that REALIZABLE nor NUMREALIZATIONS are in NP, because the said realization could be exponentially large with respect to the number of vertices or the window size. Although we cannot settle this question in general, we prove this situation occurs in the directed case (DU and DW), for which some graphs have minimal realizations whose length scales exponentially with the window size. This is formally stated in Proposition 11 for DU-REALIZABLE.

Proposition 11. *For any positive integers n and k, there exists a graph of size $3kn + 1$ such that any DU-realization with a window of size $k + 1$ has length at least $2kn^k$.*

Proof. See Figure 13 for an example. Our construction uses three sets of vertices A, B and C of size $k \times n$ each (vertices are labelled respectively $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ with $1 \leq i \leq k$ and $0 \leq i < n$), plus an additional *start* vertex s. The *column* of a vertex in $A \cup B \cup C$ is its first index, the *value* is its second index, the *rank* of a vertex v is an integer in $\mathbb{Z}/2k\mathbb{Z}$ equal to the column of v if $v \in A$ and to its column plus k if $v \in B \cup C$. Vertex s has column, value and rank 0. Given a k-tuple $T = (j_1, \ldots, j_k)$ with values in $[0, n-1]$, the *successor* of T is the k-tuple

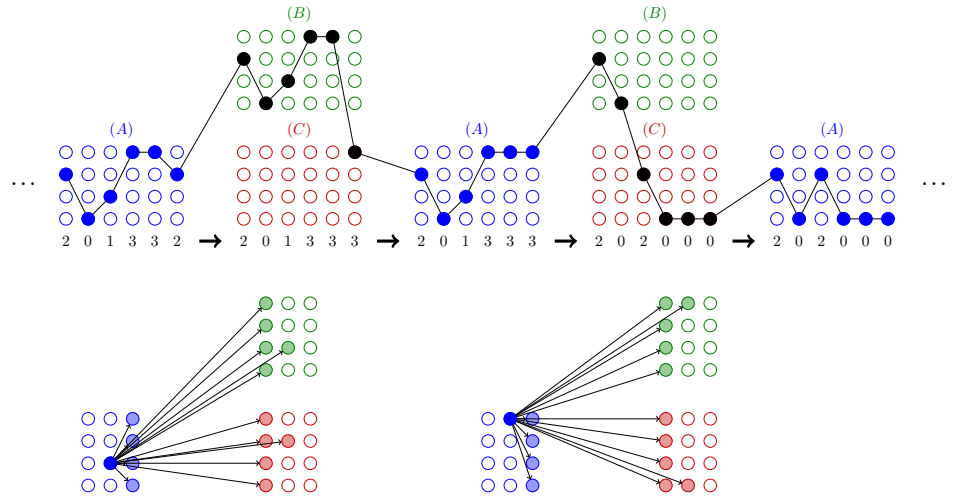


FIGURE 13. Illustration of the construction in Proposition 11 for a graph with an exponentially long realization, with $n = 4$ and $k = 6$. Top: a fragment of the path, starting with the substring $a_{1,2}a_{2,0}a_{3,1}a_{4,3}a_{5,3}a_{6,2}$: in a correct realization, such a fragment (with value $(2, 0, 1, 3, 3, 2)$) must be followed by the highlighted vertices with successive values $(2, 0, 1, 3, 3, 3)$ and $(2, 0, 2, 0, 0, 0)$. Vertices are drawn multiple times, to avoid overlappings in the drawing, but there are indeed only $n \times k$ vertices in each of A, B an C. This counting behavior must be repeated from $(0, 0, 0, 0, 0, 0)$ to $(3, 3, 3, 3, 3, 3)$, yielding a path of length at least 4^6. Bottom: example of arcs outgoing from two A vertices that enforce this behavior. Each vertex in A is connected to a single vertex in the corresponding column in each of B and C, where B is used to keep the same value and C is used to increment a column.

$T' = (0, \dots 0, j_x + 1, j_{x+1}, \dots j_k)$ where x is the smallest index such that $j_x < n - 1$, i.e. all n^k such tuples form a path from $(0, \dots, 0)$ to $(n - 1, \dots n - 1)$.

We build a DAG on vertex set $A \cup B \cup C \cup \{s\}$ with the following arcs. Vertex s has outgoing arcs to each of $a_{i,0}$ for all i. Each vertex $a_{i,j}$ with $1 \leq i \leq k$ and $0 \leq j < n$ has an outgoing arc to each $a_{i',j'}$ with $i' > i$, to each $b_{i',j'}$ with $i' < i$, to $b_{i,j}$ and to $c_{i,j+1 \mod n}$. Each vertex $b_{i,j}$ and each $c_{i,j}$ with $1 \leq i \leq k$ and $0 \leq j < n$ has an outgoing arc to each $b_{i',j'}$ with $i' > i$, to each $a_{i',j'}$ with $i' < i$ and to $a_{i,j}$. Finally, for $i < k$, each $c_{i,0}$ is connected to $c_{i+1,j}$ for all $0 \leq j < n$.

Let S be a realization of G with window size $k + 1$. Clearly S necessarily starts with s (the only vertex with in-degree 0). Let $1 \leq p \leq |S| - k$. Consider the substring $S' = S[p \dots p + k]$. Note that by construction a vertex of rank r only has outgoing arcs to vertices with rank $r + i$ with $0 < i \leq k$. In particular, two vertices of the same rank cannot be in S'. Thus, let r be the rank of $S[p]$, then all other vertices of S' have rank in $[r + 1, r + k]$. In particular, the second vertex $S[p + 1]$ in S' has out-going arcs to $k - 1$ vertices with ranks among $[r + 1, r + k]$, which is only true for vertices of rank $r - 1$, r, or $r + 1$. Thus $S[p + 1]$ has necessarily rank $r + 1$.

Hence, since $S[1] = s$ has rank 0, then $S[i]$ has rank $i-1$ for $1 \leq i \leq |S| - k$. In particular, $S[1, \ldots, k+1] = sa_{0,0} \ldots a_{k,0}$.

Let $a_{i,j} \in A$ and p such that $S[p] = a_{i,j}$. Then $S[p+k]$ is one of $b_{i,j}, c_{i,j+1}$. For $S[p] = b_{i,j} \in B$ or $S[p] = c_{i,j} \in C$ then $S[p+k] = a_{i,j}$. Thus, in most cases, the value of $S[p]$ and $S[p+k]$ are equal, except in the following cases: $S[p] = a_{i,j}$ and $S[p+k] = c_{i,j+1}$. Then by the incoming arcs of $c_{i,j+1}$, necessarily $S[p+k-i'] = c_{i-i',n}$ for all $0 < i' \leq i$. $S[p] = c_{i,n}$ and $S[p+k] = a_{i,0}$. Let p be a position such that $S[p]$ has rank 1, let $T = (j_1, \ldots, j_k)$ be the tuple of values of $S[p] \ldots S[p+k-1]$, let T' be the tuple of values of $S[p+k] \ldots S[p+2k-1]$, and T'' be the tuple of values of $S[p+2k] \ldots S[p+2k-1]$. Then if $S[p+k] \ldots S[p+2k-1]$ does not contain any vertex in C, then $T = T' = T''$. Otherwise, let x be the first index such that $j_x < n-1$, then $T' = (0, \ldots, 0, j_x + 1, j_{x+1}, \ldots, j_k)$, and $T'' = T'$ is the successor of T.

To conclude, S contains $a_{0,0} \ldots a_{k,0}$, i.e. a substring with tuple of values $(0, \ldots, 0)$. It also contains $c_{k,0}$, which has only incoming arcs from $a_{k,n-1}$ and from each $c_{i,0}$ with $i < k$, thus S also contains $a_{1,n-1} \ldots a_{k,n-1} c_{1,0} \ldots c_{k,0}$, hence S contains the tuple $(n-1 \ldots n-1)$. Since S must use consecutive tuples according to the successor relation, it must contain substrings with rank 1 to k with each tuple from $(0, \ldots, 0)$ to $(n-1 \ldots n-1)$, i.e. it has length at least $(2k)n^k$.

Note that the above proof does not guarantee the actual *existence* of such a realization. However, the construction can be adapted to this end, by providing an exponential-length sequence using only arcs from the DAG (starting with $sa_{1,0} \ldots a_{k,0}$ and ending with $a_{1,n-1} \ldots a_{k,n-1}$), and filtering out those edges that are not realized. Thus, any sequence realizing the resulting graph still requires an exponential length, and the graph is realizable by construction. □

Remark 4. *The existence of instances with exponentially large DW-realizations is due to encoding of the input, and the length p of the realization. By definition, the length of DW-realization of an instance (G, Π) depends linearly on the sum of the coefficients of Π, whereas the encoding of the entries of Π can be done logarithmically with respect to the values of Π.*

Acknowledgments

The authors wish to express their gratitude to Guillaume Fertin and an anonymous reviewer of an earlier version of this manuscript, for their valuable suggestions and constructive criticisms.

References

[1] Arora, S., Li, Y., Liang, Y., Ma, T., Risteski, A.: A latent variable model approach to pmi-based word embeddings. Transactions of the Association for Computational Linguistics **4**, 385–399 (2016)

[2] Brightwell, G.R., Winkler, P.: Counting eulerian circuits is# p-complete. In: ALENEX/ANALCO. pp. 259–262. Citeseer (2005)

[3] de Bruijn, N.G., van Aardenne-Ehrenfest, T.: Circuits and trees in oriented linear graphs. Simon Stevin **28**, 203–217 (1951)

[4] Chaiken, S.: A combinatorial proof of the all minors matrix tree theorem. SIAM Journal on Algebraic Discrete Methods **3**(3), 319–329 (1982)

[5] Gawrychowski, P., Kociumaka, T., Radoszewski, J., Rytter, W., Waleń, T.: Universal reconstruction of a string. Theoretical Computer Science **812**, 174–186 (2020)

[6] Gibert, J., Valveny, E., Bunke, H.: Dimensionality reduction for graph of words embedding. In: International Workshop on Graph-Based Representations in Pattern Recognition. pp. 22–31. Springer (2011)
[7] Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. Siam Review **56**(1), 3–69 (2014)
[8] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
[9] Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
[10] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., Yang, Q.: Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 World Wide Web Conference. pp. 1063–1072 (2018)
[11] Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
[12] Roth, M., Woodsend, K.: Composition of word representations improves semantic role labelling. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 407–413 (2014)
[13] Rousseau, F., Kiagias, E., Vazirgiannis, M.: Text categorization as a graph classification problem. In: Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 1: Long Papers). pp. 1702–1712 (2015)
[14] Sanjeev, A., Yingyu, L., Tengyu, M.: A simple but tough-to-beat baseline for sentence embeddings. Proceedings of ICLR (2017)
[15] Skianis, K., Malliaros, F., Vazirgiannis, M.: Fusing document, collection and label graph-based representations with word embeddings for text classification. In: Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12). pp. 49–58 (2018)