Sequence Graphs Realizations and Ambiguity in Language Models*

Sammy Khalife [1,2], Yann Ponty[3], Laurent Bulteau[4]

[1]Department of Applied Mathematics and Statistics, Johns Hopkins University, USA.
[2]School of Operations Research and Information Engineering, Cornell Tech, Cornell University, USA.
[3]LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, France.
[4]LIGM, CNRS, Universite Gustave Eiffel, France.

Abstract

Several popular language models represent local contexts in an input text $\boldsymbol{x}$ as bags of words. Such representations are naturally encoded by a sequence graph whose vertices are the distinct words occurring in $\boldsymbol{x}$, with edges representing the (ordered) co-occurrence of two words within a sliding window of size $\boldsymbol{w}$. However, this compressed representation is not generally bijective: some may be ambiguous, admitting 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 algorithmic point of view. We consider the existence and enumeration of realizations of a sequence graph under multiple settings: window size $\boldsymbol{w}$, presence/absence of graph orientation, and presence/absence of weights (multiplicities). When $\boldsymbol{w} = \mathbf{2}$, 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 $\boldsymbol{w} \geq \mathbf{3}$, we prove the hardness of all variants, even when $\boldsymbol{w}$ is considered as a constant, with the notable exception of the undirected unweighted case for which we propose XP algorithms for both problems. We conclude with an integer program formulation to solve the realizability problem, and a dynamic programming algorithm to solve the enumeration problem in instances of moderate sizes. This work leaves open the

*A preliminary version of this work has been published in COCOON'21.
Corresponding author: Sammy Khalife, khalife.sammy@cornell.edu

membership to NP of both problems, a non-trivial question due to the existence of minimum realizations having size exponential on the instance encoding.

Keywords: Graphs, Sequences, Combinatorics, Inverse problem, Computational Complexity

1 Introduction

The construction of numerical vector representations for words and sentences (a.k.a *embeddings*) has been a long-standing problem in data science and artificial intelligence. Until the mid 2010s, most of the techniques in information retrieval and Natural Language Processing (NLP) used pre-computed embedding as input to machine learning algorithms. In this context, the choice and computation of the "right" embedding constitutes a problem in its own right. Ultimately, each embedding of this type can be thought of as a map between words to numerical vectors: each word is attributed a *static* vector, hopefully encoding different types of information about the word (semantic, spelling, ...). In a new document, this word is mapped to the same vector, independently of the new context in which it appears. We refer the reader to [1] for a description of these embedding techniques, such as *Word2Vec*, *GloVe*, *FastText*, and others. These embeddings laid the groundwork for modern NLP techniques, by being scalable, interpretable, and performing for several information retrieval tasks. On the other hand, attention-based models [2], and its various implementations such as *transformers* [3], adopt a different paradigm by incorporating both the embeddings and the learning parameters in the same computational framework. Furthermore, they allow the embedding of each word to *depend* on the context it appears in. This new paradigm combined with training techniques lead to state-of-the-art methods in most NLP tasks including complex ones that traditional embeddings were not able to handle well, in particular translation, summarization, question answering, and natural text understanding.

Measuring the capacity of embeddings to faithfully represent words or sentences is a natural problem in order to understand their limitations. In particular, one may wonder and ask about their level of *ambiguity*: *How many changes can be made to a sentence that would yield the same embedding? How long does a sequence need to be such that the resulting embedding is not unique?* More generally, the question of invariance (and equivariance[1]) in computational linguistics – including embeddings – has become a topic of growing interest: in a recent line of research initiated by [4, 5], the authors introduce the idea that *equivariance* in embeddings and can also be a desirable property. Namely, Gordon et al. [4] exposed a method to construct *string equivariance*, which suppose some form of equivariance under "local" permutation action. White et al. [6] further generalize this work by restricting the range of permutations only to some lexical classes: symmetries are only allowed to exchange words inside a given fixed lexical class (examples of lexical classes are subsets of verbs, adjectives, nouns

[1] Informally, embeddings are equivariant if under some group action, changes in the input sequence should affect the embeddings in the same manner.

of interchangeable semantic roles). This line of work, builds on the premise that certain embeddings exhibit desirable properties, enabling them to emulate fundamental cognitive abilities such as *compositional generalization*[2] [7–9]. In turn, embeddings sharing such properties would lead to general computational frameworks coming closer to a true understanding of words and sentences [10].

In this article we study a family of combinatorial problems related to the first group of *context-based* embeddings (including *Word2Vec*, *Glove*, *FastText*, ...). Some of these problems can be naturally interpreted as determining the level of *ambiguity* of these embeddings. While context-based embeddings are richer than the traditional *bag-of-words* (related to *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). The main contribution of this article is to present new complexity results about the inverse problems quantifying such level of ambiguity. We also present algorithms that allow to solve these problems of graphs of reasonable size, despite the computational hardness in most variations of the problems. Our results also highlight the potential limitations of context-based embedding to encode the whole information of sequences of reasonable length. In the line of recent efforts on invariance and equivariance in computational linguistics, we hope that this study can be connected to intrinsic desirable properties of word embeddings, and serve as a stepping stone to further understand their attention-based variants [5].

1.1 Definitions and problem statement

In the following, p is a positive integer and $[p]$ is a shorthand for $\{1, ..., p\}$. Let $x = x_1, x_2, ..., x_p$ be a finite sequence over a vocabulary $X = \{v_1, \cdots, v_n\}$. We first formalize the notion of *sequence graph* – introduced in [11] as *graph-of-words* – illustrated in Figures 1 and 2.

Definition 1. *Given a sequence x and a window size $w \in \mathbb{N}^+$ ($w > 0$), the* sequence graph of x with window size w *is the graph where each vertex of $G = (V, E)$ is the set of distinct words appearing in the sequence, and each edge notifies the appearance of two words in a context of size w. Formally, $V = \{v \in X \mid \exists i \in [p],\ v = x_i\}$ and*

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

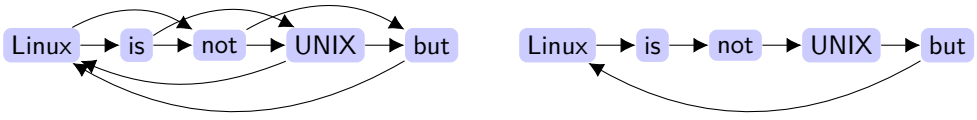
A weighted sequence graph G is endowed with a weight matrix $\Pi(G) = (\pi_e)_{e \in E}$ such that

$$\pi_{\{u,v\}} = \mathsf{Card}\ \{(k, k') \in [p]^2 \mid\ 0 < |k - k'| < w,\ x_k = u \text{ and } x_{k'} = v\} \tag{2}$$

For digraphs, the two-element set $\{u, v\}$ is replaced by the pair (u, v) in Statement (1), and the absolute values in Statements (1) and (2) are replaced with $k < k' < k + w$.

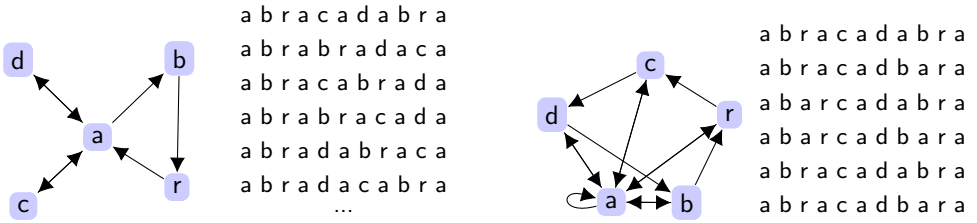
Under these conditions, we say that x is a w-realization *of G, or* realization *in the absence of ambiguity, if G is the sequence graph of x with window size w. Finally, x is a* w-realization *of (G, Π) if G is the w-sequence graph of x with $\Pi(G) = \Pi$.*

[2] In a nutshell, compositional generalization is the ability to process a novel sentence and assign overall possible meanings (e.g. "cook twice") by composing the meanings of its individual parts (e.g. meanings of "cook" and "twice").

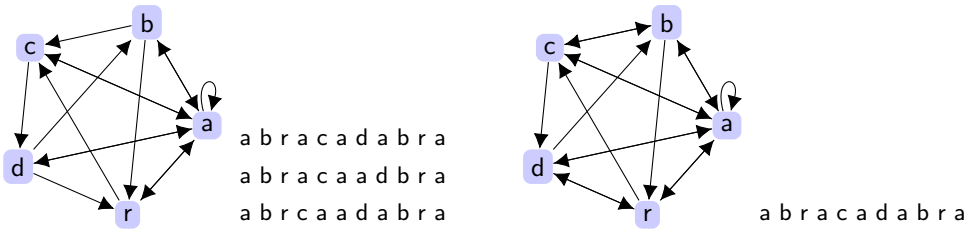


(a) Unambiguous graph ($w = 3$) (b) Ambiguous graph ($w = 2$)

Fig. 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

Fig. 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 and the natural integers $\pi_{\{u,v\}}$ represent the number of co-occurrences of u and v in all windows of size w. An algorithm to construct a weighted sequence graph is presented in Algorithm 1; the other cases (unweighted, directed) 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) admits a w-realization x, False otherwise.*

Algorithm 1 Construction of Π associated to a weighted sequence graph

Parameter: Window size $w \geq 2$
Input: Sequence x of length $p \geq 1$ of integers in $[1, n]$
Output: Weighted adjacency matrix $\Pi = (\pi_{\{u,v\}})$

```
π_{u,v} ← 0 for u, v ∈ [n]^2
for i = 1 to p − 1 do
    for j = i + 1 to min(i + w − 1, p) do
        π_{x_i,x_j} ← π_{x_i,x_j} + 1
return Π = (π_{u,v})
```

Problem 2 (Unweighted-Realizable (U-Realizable))**.**
Input: *Graph G (directed or undirected), window size w*
Output: *True if G admits a w-realization x, False otherwise.*

We denote by DW-Realizable (resp. GW-) the restricted version of W-Realizable where the input graph G is directed (resp. undirected). We define GU-Realizable and DU-Realizable similarly. Our notations can be summarized as:

DW $\rightarrow$ (directed) Digraph, Weighted	DU $\rightarrow$ (directed) Digraph, Unweighted
GW $\rightarrow$ (undirected) Graph, Weighted	GU $\rightarrow$ (undirected) Graph, Unweighted

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 i.e. $|\{x \in X^\star \mid \phi_w(x) = G\}|$ if 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 (DW,DU, GW, GU). We also denote NumRealizations$_w$ for the case where w is a fixed positive integer. Note that NumRealizations generalizes the previous one, as Realizable can be solved by testing the nullity of the output of NumRealizations.

1.2 Related work

Sequence graphs encode the information in several models based on co-occurences [11–13]. 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 are similar to the *Universal Reconstruction of a String* [14], which consists in determining the set of strings of a fixed length having as many distinct letters as possible, satisfying substring equations of the form: $s[q_1 \cdots q_p] = s[q'_1 \cdots q'_p], \cdots, s[r_1 \cdots r_m] = s[r'_1 \cdots r'_m]$, where $s[q_1 \dots q_p]$ refers to the substring $s_{q_1} \dots s_{q_p}$ and the increasing indices q_i's, q'_i's, $\cdots$, r_i's and

r'_i's, and the length of s are given as part of the input. The reconstruction problem consists in finding a string s that verifies the given set of constraints, with a maximum number of distinct letters. We shall see that these problems are intrinsically different; in particular, the complexity results presented in this article imply the absence of reduction to the reconstruction problem solvable in linear time with respect to the length of the input string s. In a similar fashion, *De Bruijn* graphs [15] are directed graphs representing overlaps between sequences of symbols. They can be seen as a specialization of the problem in this paper, using window size 2. Given a positive integer k, the vertices of a De Brujin graph formed from a sequence are the k-mers (sub-sequence of k symbols) in the sequence, and edges are formed between a pair k-mers when one appears just before another (two adjacent k-mers will share $k-1$ consecutive symbols: as a suffix for the first one, and as a prefix for the other). The main difference with our setting is that distinct size-k windows are encoded into distinct nodes even if they share many symbols (in different orders), while our setting has a very sparse set of nodes (one per symbol) and all context information is held by edges.

Inverse problems studied in the Distance Geometry (DG) literature also bear some similarities. 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 [16] (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 in the DG instance. In sequence graphs 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 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.

1.3 Paper outline

In the following section we give a complete overview of our structural and algorithmic results: first for window size 2 (Section 2.1), then for larger window sizes (Section 2.2). We prove results for window size 2 in Section 3, and then give detailed algorithms and hardness reductions for larger windows in Section 4. More precisely, we first focus on the GU variant (with an XP algorithm and W[1]-hardness proof) in Section 4.1, then hardness proofs for other variants in Section 4.2, and finally exponential-time algorithms for weighted variants (GW and DW) in Section 4.3. Finally, We present a construction yielding exponential-size realizations in Section 5.

Table 1: Complexity of variants of NumRealizations and Realizable with $w = 2$. The possible number of sequences is given for each variant of NumRealizations, and a characterization of yes-instances is given for Realizable.

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

2 Results overview

In this section, we present our main theoretical results for $w = 2$ in Subsection 2.1 and for $w \geq 3$ in Subsection 2.2. For the sake of readability, we postpone full proofs of our results to Sections 3 and 4 respectively. Overall, our results reveal a stark contrast between $w = 2$, where all relevant problems can be solved in polynomial time, and $w \geq 3$ where most versions of our problems become hard, except for GU which admits a slicewise polynomial (XP) algorithm parameterized by w.

2.1 Complete characterization of 2-sequence graphs ($w = 2$)

A graph has a 2-realization 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 times for each edge e in the weighted case). This observation enables relatively simple characterization and algorithmic treatment, leading to the results summarized in Table 1 and Theorem 1. The additional definitions are given below.

Definition 2 ($\psi(G)$)**.** *Let (G, Π) be a weighted graph (directed or undirected). $\psi(G)$ is the multigraph with the same vertices as G and with multiplicity π_e for each edge $e \in E$.*

Definition 3 ((semi-)Eulerian)**.** *We say that a path is* (semi-)Eulerian *if it visits all edges of the graph exactly once, and a graph is (semi-)Eulerian if it admits a (semi-)Eulerian path. A (semi-)Eulerian path with identical endpoints is a* Eulerian cycle*, otherwise it is a* semi-Eulerian path*, and this distinction extends to Eulerian and semi-Eulerian graphs (here this distinction is only made in Proposition 4).*

Definition 4 ($R(G)$, $R^+(G)$)**.** *Let $G = (V, E)$ be a digraph. $R(G)$ is the Directed Acyclic Graph (DAG) such that: i) every strongly connected components of G is associated to a unique node in $R(G)$, and ii) two strongly connected components $u \neq v$ in G form an edge (u, v) in $R(G)$, provided there exists an edge $(x, y) \in E$ such that $x \in u$ and $y \in v$.*

$R^+(G)$ is the weighted DAG such that: i) $R^+(G)$ has the same vertices and edges as $R(G)$ and ii) the weight of an edge in $R^+(G)$ is the number of distinct edges between two strongly connected components in G.

Definition 5 (simple step graph)**.** *Let G be a digraph. G is said to be a* simple step graph *(see Figure 3) if $R^+(G)$ is a directed path and its edges have weight 1.*

(a) G is a simple step, $R^+(G)$ contains a single node

(b) G is not a simple step, $R^+(G)$ is a path with a weight-2 edge

Fig. 3: Illustration of Definition 4 of graph $R^+(G)$ (drawn with brown edges and background) and Definition 5 of a simple step graph. The graph in (a) has a single strongly connected component (yielding a trivial graph $R^+(G)$). The graph in (b) has two strongly connected components connected with two distinct edges ((1,3) and (2,4)), so they form a weight-2 edge in $R^+(G)$.

Theorem 1. *A weighted graph G (directed or undirected) admits a 2-realization if and only if $\psi(G)$ is (semi-)Eulerian. An unweighted undirected (resp. unweighted directed) graph admits a 2-realization if and only if it is connected (resp. simple step).*

All variants of Realizable$_2$ *are in P. For* NumRealizations$_2$*, the* GW *variant is #P-hard, and all others are in P.*

2.2 General complexity and algorithmic results ($w \geq 3$)

In this subsection we present the remaining complexity results, which are summarized in Theorem 2 and Table 2. We first show that GU-Realizable$_w \in P$ for any integer $w \geq 3$. Besides, for GU, the number of realizations of a graph G is either 0 (not realizable), 1 or $+\infty$ (realizable in both cases). These three cases can be tested in polynomial time using our algorithm (presented in Section 4), showing that GU-NumRealizations$_w \in P$, for any integer $w \geq 3$. All proofs of the following statements are given in Section 4.

Theorem 2. *For any integer $w \geq 3$, the* GW*,* DU *and* DW *variations of* NumRealizations$_w$ *and* Realizable$_w$ *are NP-hard, and the* GU *variations are in P.*

In other words, NumRealizations and Realizable parameterized by w are para-NP-hard in the GW, DU and DW variations and slicewise polynomial (XP) in the GU variations.

We further investigate the parameterized complexity for GU-Realizable, since an XP algorithm, here in time $O(n^w)$, can sometimes be improved into an FPT algorithm running in time $f(w)n^{O(1)}$. This is however, not the case here (under usual complexity assumptions), as we prove the parameterized hardness of this problem.

Theorem 3. GU-Realizable *is* W[1]*-hard for parameter w.*

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

Finally, we describe in Section 4.3 an integer linear formulation of DW and GW-Realizable, and a $\mathcal{O}(n^w 2^{w\,p})$ dynamic programming algorithm for NumRealizations.

3 The special case of window size 2 ($w = 2$)

In this section we present the proofs of the results gathered in Table 1 and Theorem 1. Apart from the GU variant (Lemma 1), 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. Definition 5, Lemma 2 and Corollary 1). The weighted cases (GW and DW) are treated with direct reductions to the problem of existence and counting of (semi-)Eulerian paths in a graph (Lemma 3).

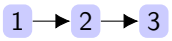
3.1 Unweighted realizability in undirected (GU) and directed (DU) graphs

The following two propositions are obtained using a simple greedy algorithm: starting from any vertex of the graph, if an edge is not realized by a candidate sequence, append any path from the current last vertex of the sequence to the missing edge, and repeat until all edges are realized.

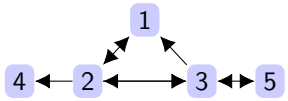
Lemma 1 (GU characterization)**.** *If $G = (V, E)$ is unweighted and undirected, with $|V| > 1$, the following are equivalent:*
(i) G is connected
(ii) G has a 2-realization
(iii) G admits an infinite number of 2-realizations.
In these conditions, a 2-realization can start and end at any vertex.

Proof (i)⇒ (ii) is obtained with the greedy algorithm described above. For (ii)⇒(iii), simply notice that any 2-realisation ending with ab can be continued with arbitrarily many additional repeats of ab. For (iii)⇒(i), a path between any two vertices can be found using any substring of any realization starting and ending with the two given vertices. □

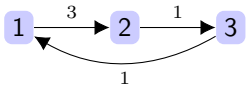
The previous characterization does not extend to strongly connected digraphs. However, we do get the following sufficient conditions, which are not necessary, as seen



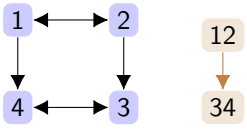
(a) 1 2 3 is a 2-realization but G is not strongly connected



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



(d) G (left) is not a 2-sequence graph whereas $R^+(G)$ (right) is one

Fig. 4: Some special cases for $w = 2$, acting as counterexamples for variants of Propositions 1 and 3.

in Figures 4a and 4b. Furthermore, they only apply in the unweighted case, as shown in Figure 4c.

Proposition 1. *Let $G = (V, E)$ be an unweighted digraph.*

(i) If G is strongly connected then G has a 2-realization. A 2-realization can start or end at any given vertex of G.

(ii) If G is (semi-)Eulerian then G has a 2-realization.

Proof Condition (i) is obtained using the greedy algorithm described above in the proof of Lemma 1.

Condition (ii) follows from the definition: a (semi-)Eulerian path is a 2-realization in the context of unweighted graphs. □

For the special case of DAGs, we have the following simple characterization of 2-sequence graphs.

Proposition 2. *If $G = (V, E)$ is a DAG, then it is a 2-sequence graph if and only if it is a directed path, in which case G has a unique 2-realization.*

Proof The backward direction ($\Leftarrow$) is an application of Proposition 1(ii): directed paths admit a unique semi-Eulerian path, that yield a unique 2-realization.

For ($\Rightarrow$), note first that a DAG is a path if and only if it is connected with in- and out-degree at most 1. Let us suppose G is a 2-sequence graph and is not a directed path, then we show that it contains a cycle. Indeed, there exists a vertex v having either two children, or two parents. In the first case, denote c_1 and c_2 for the two distinct children of v. Then there exists a walk going through both (v, c_1) and (v, c_2) (w.l.o.g. in this order), so G also has a path from c_1 to v, which creates a cycle with (s, c_1). Similarly, if v has two parents p_1 and p_2, then any 2-realization yields a walk through both (p_1, v) and (p_2, v), and thus a cycle. □

We now turn to general digraphs and give a necessary conditions for G to admit a 2-realization (which is not sufficient, as shown using an example in Figures 4d).

Proposition 3. *Let $G = (V, E)$ be 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 for the sake of contradiction let us suppose that $R(G)$ is not a 2-sequence graph. Since $R(G)$ is a (weakly) connected DAG, then using Proposition 2, it cannot be a directed path, so $R(G)$ has either a node having two children or two parents. Let us assume without loss of generality that $R(G)$ has a node v in $R(G)$ with two children c_1 and c_2. Recall that v, c_1 and c_2 represent three distinct connected components of G (in particular three sets of vertices in G that have empty intersection). Hence, there exist $v_1, v_2 \in V, w_1 \in c_1$, and $w_2 \in C_2$ such that $(v_1, w_1) \in E$ and $(v_2, w_2) \in E$. Consider now the 2-realization of G, assuming without loss of generality that (v_1, w_1) is realized before (v_2, w_2). Then there exists a path between w_1 and v_2 in G, which implies that w_1 belongs to the same component as v_1 and v_2: a contradiction. □

Based on the sufficient and necessary conditions above, we can now converge to a complete characterization of 2-sequence graphs in the DU setting:

Lemma 2 (DU characterization)**.** *Let $G = (V, E)$ be an unweighted digraph. G is a 2-sequence graph if and only if it is a simple step graph. This property can be verified in linear time.*

Proof If G is a 2-sequence graph, $R(G)$ is a 2-sequence graph using Proposition 3. Proposition 2 implies that $R(G)$ and $R^+(G)$ are directed paths. Moreover, if $R^+(G)$ has an edge with weight 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 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 $\mathcal{P} = (c_1, ..., c_p)$ such that:

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

(ii) For any $i \in \{1, \cdots, p-1\}$, there exists a unique edge (u, v) of G with $u \in c_i$ and $v \in c_{i+1}$.

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

Base case: c_1 is a strongly connected component of G, we initialize y with any 2-realizations of c_1 (which exists by Proposition 1(i)).

For $i \in \{1, .., p-1\}$: let $e = (v, w)$ be the single edge with $v \in c_i$ and $w \in c_{i+1}$. By construction, all the edges induced by c_i have already been added to y. Suppose at the previous step the last vertex added is $z \in c_i$. We first add all vertices of a walk starting at z (excluded) and ending at v. Then, consider a walk starting at w (included) and which visits every edge of c_{i+1} (again using Proposition 1(i)). We add all vertices of this walk after v.

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

Finally, note that verifying if a graph is simple step is linear, as it directly follows from a strongly connected components decomposition. □

A direct consequence of Lemma 2 is the following:

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

Proof First, if G is a DAG, then by Proposition 2 it has either zero or a unique 2-realization (exactly one if it is a directed path). If G is not a DAG, G has a cycle $u_0u_1 \dots (u_\ell = u_0)$ (possibly with $\ell = 1$ in case of self-loops) and admits a 2-realization y. Then y has at least one occurrence of u_0, and a strictly longer 2-realization y' can be obtained by inserting $u_0 \dots u_{\ell-1}$ just before any occurrence of u_0. Therefore G has infinitely many 2-realizations. □

3.2 Weighted realizability in undirected (GW) and directed (DW) graphs

The weighted cases (GW and DW) cannot be treated similarly due to the weight constraints implying that a weighted graph has a finite number of realizations (as was seen in Figure 4c). However, in this setting, we can use (semi-)Eulerian paths to obtain the desired result.

Theorem 4. *If G is a weighted graph (possibly directed), with weight matrix $\Pi(G)$, 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 3. *Let $G = ((V, E), \Pi)$ be a weighted 2-sequence graph (possibly directed). 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 from 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_e copies of each edge e 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 (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 seen as 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 $\pi_e > 1$, 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 $\sim$ on $\mathcal{E}$: For two (semi-)Eulerian paths P_1 and P_2, $P_1 \sim P_2 \iff P_1$ can be obtained from P_2 by permuting edges of $\psi(G)$ that are copies of the same edge in G. $\sim$ is an equivalence relation because it is symmetric, transitive and reflexive. Let $\mathcal{E}/\sim$ be $\mathcal{E}$ quotiented by $\sim$. We have $P_1 \sim P_2 \iff f(P_1) = f(P_2)$ (equivalently, P_1 and P_2 yield the same sequence of vertices), so $|\mathcal{S}|$ is the number of equivalence classes of $\sim$, or equivalently, $|\mathcal{E}/\sim|$. Note that each equivalence class of $\sim$ 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 $|\mathcal{S}| = |\mathcal{E}/\sim| = |\mathcal{E}|(\prod_{e\in E} \pi_e)^{-1}$. □

On the one hand, counting the number of (semi-)Eulerian paths in a undirected graph is a $\#P$-complete problem [17]. Since $G \mapsto \psi(G)$ is bijective, counting the number of 2-realizations is also $\#P$-complete in the GW setting. On the other hand, for the DW setting, counting (semi-)Eulerian paths of a weighted digraph is in P, and can be derived using the following proposition (writing $\deg_{\psi(G)}(v)$ for the indegree of a vertex v in $\psi(G)$, i.e. $\deg_{\psi(G)}(v) = \sum_{u\in V} \pi_{(u,v)}$):

Proposition 4. *Let $G = (V, E)$ be a weighted digraph, with $\Pi(G)$ an $n \times n$ matrix of integers. Then, 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 and $Sp(L)$ the set of eigenvalues of L, then

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

- 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 $\tilde{\psi}(G) := \psi(G) + (u, v)$, and divide the output by the number of vertices $|V|$.

Proof The case of $\psi(G)$ being Eulerian is a direct consequence of Lemma 3, BEST Theorem [18] and Matrix Tree Theorem [19].

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 $\psi(G) + (u, v)$). □

Corollary 2. *Let G be a weighted graph (directed or undirected). For every non-negative integer n, there exists a sequence graph having n 2-realizations.*

Proof Let $n \geq 0$ be an integer. The case $n = 0$ and $n = 1$ are trivial, so we suppose now that $n > 1$. In the directed case, simply consider the oriented cycle C_n on n vertices where all edges have weight 1. Then, C_n has exactly n realizations where each sequence is determined by one of the n starting vertices.

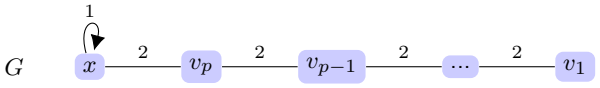
In the undirected case, if n is even, the (undirected) cycle with unit weights $C_{\frac{n}{2}}$ gives n 2-realizations (both directions are now allowed). If $n = 2p + 1$ with $p \in \mathbb{N}$, consider the sequences defined as follows and their reverse:

$$v_1 v_2 \cdots v_p \, x x \, v_p \cdots v_2 v_1 \tag{1}$$

$$v_{i+1} v_i \cdots v_2 v_1 v_2 \cdots v_p \, x \, x \, v_p \cdots v_{i+1} \quad \forall i \in [p-1] \tag{2}$$

$$x \, v_p v_{p-1} \cdots v_2 v_1 v_2 \cdots v_p \, x \, x \tag{3}$$

This represents a total of $1 + 2(p-1) + 2 = n$ sequences (since (1) is its own reverse). First observe that all these sequences yield the same sequence graph G, shown below



Thus G admits at least n 2-realizations. It remains to show that every 2-realization of G is of one of the forms (1), (2), (3) above. This is based on the observation that the vertex x must appear twice or three times.

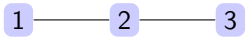
If x appears exactly twice, it has to appear next to itself and next to v_p twice, so the sequence contains the subsequence $v_p \, x \, x \, v_p$. We are exactly in case (1) or (2) as the rest of the sequence has to verify the adjacency constraints between the v_i's.

If x appears exactly three times, then it has one occurrence as the first element and another as the last element. Moreover, it has to appear next to itself as $x \, x$ (either for the first or last occurrence). The rest of the sequence is then entirely determined by the adjacency constraints and we are in case (3). □

4 Complexity and algorithms for general window sizes ($w \geq 3$)

The characterization of general sequence graphs differs from the one of 2-sequence graphs: the undirected graph in Figure 5a satisfies the conditions of Lemma 1 but has no 3-realization, and similarly in a directed setting the graph in Figure 5b satisfies the conditions of Lemma 9 but is not 3-realizable.

In fact, there is no simple characterization of realizable graphs for larger window size, and most variants of Realizable are already NP-hard for window size 3 (Section 4.2). However, we do get a polynomial-time algorithm for the GU variant for any fixed window size that we present first in Section 4.1.1, matched with a parameterized complexity lower bound shown in Section 4.1.2.



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

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

Fig. 5: Non-realizable graphs for window size 3. Indeed, any 3-realization of length at least 3 in an undirected (resp. directed) graph yields either a self-edge (resp. self-loop) or a clique (resp. tournament) of size 3, and these graphs have neither.

4.1 Parameterized results for undirected unweighted graphs (GU)

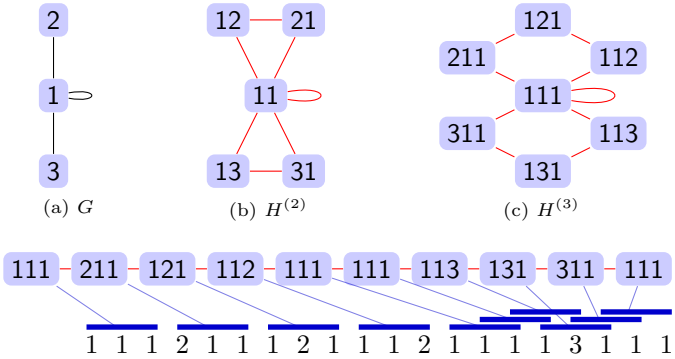
For undirected unweighted graphs (GU), we describe the construction of a size-n^w auxiliary graph reducing the question of realizability to mere connexity, thus giving a slicewise polynomial (XP) algorithm for the w parameter. We then show that such n^w factor in the complexity is unavoidable due to a W[1]-hardness reduction from Clique.

4.1.1 Slicewise polynomial (XP) algorithm for GU-Realizable

We introduce the following auxiliary graph used in our polynomial time algorithm for GU-Realizable$_w$. See Figure 6 for an example.

Definition 6. *Let $G = (V, E)$ be an undirected graph and $k \geq 2$. We write $v_{1:k}$ as a shorthand for a length-k string of nodes $v_1 \dots v_k$. Let $H^k(G) = (V^{(k)}, E^{(k)})$ be the undirected graph with*

$$V^{(k)} = \{v_{1:k} \mid \forall 1 \leq i < j \leq k, \{v_i, v_j\} \in E\}\}$$
$$E^{(k)} = \{\{u_{1:k}, v_{1:k}\} \mid u_{2:k} = v_{1:k-1} \text{ and } \{u_1, v_k\} \in E\}$$



(a) G (b) $H^{(2)}$ (c) $H^{(3)}$

(d) A walk visiting all arcs of (the single connected component of) $H^{(3)}$ and its underlying string that is a 4-realization of G by Lemma 5.

Fig. 6: Top: example of construction of the auxiliary graphs $H^{(k)}$ for $k = 2$ and $k = 3$. Bottom: conversion of a walk in $H^{(3)}$ into a 4-realization of G.

We now show that finding a w-realization of G is equivalent to finding a connected component of the auxiliary graph that *covers* all edges, as defined below. With this step we remove the need to consider long permutations of vertices, thus reducing the combinatorics to the size of H (that is, $n^{O(w)}$).

Definition 7. *An edge $\{x, x'\}$ of G is* covered *by a vertex $y \in V^{(k)}$ if $x = y_i$ and $x' = y_j$ for some $1 \leq i < j \leq k$; it is covered by an edge $\{y, y'\}$ in $E^{(k)}$ if $y = x_{1:k}$, $y' = x'_{1:k}$ with $x = x_1$, $x' = x'_k$ and $x_{2:k} = x'_{1,k-1}$. Edge $\{x, x'\}$ is covered by a subgraph of $H^{(k)}$ if it is covered by at least one vertex or one edge in this subgraph.*

Lemma 4. *If G has a w-realization, then $H^{(w-1)}$ has a connected component covering all edges.*

Proof Write $k = w - 1$. Let $P = x_{1:\ell}$ be a w-realization of G. For each $i \in [\ell - k + 1]$, let $y_i = x_{i:i+k-1}$. Since P is a w-realization, we have $\{x_p, x_q\} \in E$ for each $i \leq p < q \leq i + w - 1$, so y_i is a vertex of $V^{(k)}$ for each $i \in [\ell - k + 1]$ and (y_i, y_{i+1}) is an edge of $E^{(k)}$ for each $i \in [\ell - k]$. Thus, $V' = \{y_1, \ldots, y_{\ell-k+1}\}$ induces a connected subgraph of $H^{(k)}$. Moreover, each edge $\{x, x'\}$ is realized in P with $x = x_i$ and $x' = x_j$, $i+1 \leq j \leq i+w-1$. We distinguish three cases: (a) if $j \leq i + k - 1 = i + w - 2$ and $i \leq \ell - k + 1$, then $\{x, x'\}$ is covered by vertex $y_i \in V'$. (b) if $j = i + w - 1$, then $\{x, x'\}$ is covered by $\{y_i, y_{i+1}\}$ in $H^{(k)}[V']$. (c) if $i > \ell - k + 1$, then $\{x, x'\}$ is covered by vertex $y_{\ell-k+1} \in V'$.

Overall, $\{x, x'\}$ is covered by the connected subgraph $H^{(k)}[V']$, so some connected component of $H^{(k)}$ covers all edges of G. □

The following definition gives the main algorithmic tool to build a w-realization from any walk in the auxiliary graph.

Definition 8. *Let $\{y, y'\} \in E^{(k)}$ with $y = x_{1:k}$, $y' = x'_{1:k}$ and $x_{2:k} = x'_{1:k-1}$. The* addition $a_y(y')$ of y' with respect to y *is the single-character string x'_k. The* addition $a_{y'}(y)$ of y with respect to y' *is the length-k string $x_{1:k}$. Given a walk $P = (y_1, \ldots, y_\ell)$ of $H^{(k)}$, the* underlying sequence of P, *denoted S_P is the concatenation $y_1 \cdot a_{y_1}(y_2) \cdot a_{y_2}(y_3) \cdots a_{y_{\ell-1}}(y_\ell)$.*

Lemma 5. *For a path P, the underlying sequence of P realizes (with window size $k+1$) exactly the edges covered by the subgraph $H^{(k)}[P]$.*

Proof We first prove the following claim: for any $\{y, y'\} \in E^{(k)}$, the string $y \cdot a_y(y')$ ends with y', and realizes exactly the edges covered by y, y' and $\{y, y'\}$ with window $k+1$.

Let $x_{1:k} = y$ and $x'_{1:k} = y'$. We distinguish the *forward* case with $x_{2:k} = x'_{1:k-1}$ from the *backward* case with $x'_{1:k-1} = x_{2:k}$.

First note that string $y \cdot a_y(y')$ starts with y (by construction) and ends with y' (this is clear in the backward case, and follows from $x_{2:k} = x'_{1:k-1}$ in the forward case). Thus, the first size-k window of $y \cdot a_y(y')$ realizes edges $\{x_i, x_j \mid 1 \leq i < j \leq k\}$ which are exactly edges covered by y. Similarly, the last size-k window of $y \cdot a_y(y')$ realizes edges $\{x'_i, x'_j \mid 1 \leq i < j \leq k\}$ which are exactly edges covered by y'.

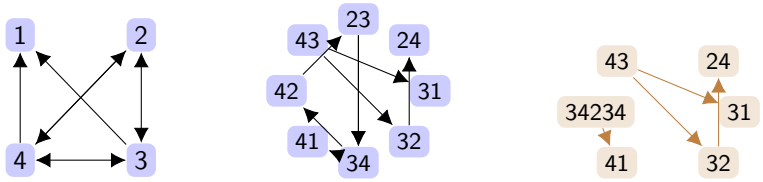
We now enumerate the remaining edges realized by size-$(k+1)$ windows of $y \cdot a_y(y')$. In the forward case, $\{x_1, x'_k\}$ is the only such remaining edge, and it is the one covered by $\{y, y'\}$. In the backward case, the remaining edges are those of the form $\{x_i, x'_j\}$ with $1 \leq j < i \leq k$. For $i = k$ and $j = 1$, $\{x_i, x'_j\}$ is the edge covered by $\{y, y'\}$. For $j > 1$, $\{x_i, x'_j\} = \{x_i, x_{j-1}\}$ is covered by y, and finally for $i < k$, $\{x_i, x'_j\} = \{x'_{i+1}, x'_j\}$ is covered by y'. Overall, the edges realized by all size-$(k+1)$ windows of $y \cdot a_y(y')$ are exactly those covered by y, y' and $\{y, y'\}$. This completes the proof of the claim.

Now for the Lemma statement, let $P = (y_1, \ldots, y_\ell)$ and S_P be the underlying sequence of P. Note that S_P contains all $y_i a_{y_i}(y_{i+1})$ as substrings by construction and each size-$(k+1)$ widows of S_P appear as a size-$(k+1)$ window of some $y_i a_{y_i}(y_{i+1})$. Thus S_p realizes (with window size $k+1$) exactly the edges realized by some $y_i a_{y_i}(y_{i+1})$, which in turn are exactly the edges covered by vertices y_i, $1 \leq i \leq \ell$ and edges $\{y_i, y_{i+1}\}$, i.e. edges covered by the subgraph $H^{(k)}[P]$. □

Lemma 6. *A graph $G = (V, E)$ is w-realizable if and only if $H^{(w-1)}$ has a connected component covering all edges. Moreover, a realization (if any) can be computed in time $O(n^w)$.*

Proof The forward direction is proven in Lemma 4. The reverse follows from Lemma 5: pick such a connected component of $H^{(w-1)}$, and let P be a walk covering all edges of this component: the underlying sequence of P is a w-realization of G. Walk P can be computed with a DFS, by visiting all edges incident to each successive node, which can be performed linearly with respect to the number of edges in H. There are at most n^w such edges (with $n = |V|$) since each edge involves at most w distinct vertices of G, so we obtain the desired time bound. □

Remark 1. *For digraphs, the aforementioned procedure can easily be translated using a directed definition of graph $H^{(k)}$, where edge $\{u_{1:k}, v_{1:k}\}$ is replaces by arc $(u_{1:k}, v_{1:k})$*



(a) G (b) $H^{(2)}$ (c) $R(H^{(2)})$

Fig. 7: Adaptation of the algorithm for GU-Realizable$_w$ to DU-Realizable$_w$, here with $w = 3$ (see Remark 1). The walk 34234, 41 in $R(H^{(2)})$ gives a 3-realization: 3 4 2 3 4 1. However, finding such a walk takes exponential time in the directed case.

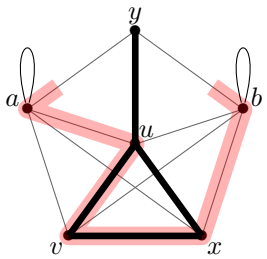


Fig. 8: Illustration of the reduction for Theorem 3, with $w = 4$. The source graph G has vertices $\{u, v, x, y\}$ with the bold edges. Vertices a and b and thin edges are added in the reduction. A realization follows a path visiting both vertices a and b: the first w vertices of the transition between a and b (highlighted in red) must form a clique in the graph, yielding a $(w-1)$-clique in the original graph.

in Definition 6. See Figure 7. One then needs to produce a directed walk visiting strongly connected components and covering all edges of G. However, there can be exponentially many walks between components in the auxiliary graph, and the choice of which components to visit leads to an exponential blow-up in the complexity of the problem.

4.1.2 Complexity lower bound for parameter w

We now prove Theorem 3, restated below, using a polynomial time parameterized reduction from Clique (decide whether a graph contains k vertices inducing all possible $\binom{k}{2}$ edges). See Figure 8 for an illustration.

Theorem 3. GU-Realizable *is* W[1]*-hard for parameter* w*.*

Proof Let $G = (V, E)$ be a simple graph. Let G' be a graph constructed from G by adding two nodes a and b with self loops, such that a and b are connected to each vertex of G. Let k be a positive integer and $w = k + 1$. We will show that G has a k-clique if and only if G' is 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 $i_b - i_a \geq w$, since there is no 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 of directed and weighted variants for constant window size $w \geq 3$

We prove in this section the NP-hardness of the remaining three variants (DU, GW, DW) for constant window size greater than 3.

Proposition 5. 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 7 below). We then give the reduction for the unweighted case (see Lemma 9 in Section 4.2.1), for which we introduce an intermediate variant with *optional* arcs. Finally we give a reduction for both weighted cases in Section 4.2.2, using the same construction for both directed and undirected cases (simply ignoring arc orientations in the latter case, see Lemma 10).

We consider slightly constrained versions of Hamiltonian Path where we require that the input graph contains up to two degree-one vertices. More formally, we reduce from the following restrictions, which are known to be NP-hard (these are folklore results, included here for completeness):

Lemma 7. Hamiltonian Path *is NP-hard even on the graph classes defined by the following 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 reduction to HP1 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 v_1 is a source vertex and any cycle in the original graph is equivalent to a path from v_1 to v_2 in the new graph.

The reduction to HP2 is from Hamiltonian Cycle in undirected graphs: pick any vertex v and duplicate it into v_1, v_2. Each edge $\{u, v\}$ becomes two edges $\{u, v_1\}$ and $\{u, v_2\}$. Add pending vertices s and t connected to v_1 and v_2 respectively. Then any cycle in the original graph is equivalent to a path in the new graph with $\{s, v_1\}$ at one end and $\{t, v_2\}$ at the other. □

4.2.1 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.*
Output: *True if there is 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; False otherwise.*

Note the following similarity between OptionalRealizable$_2$ and Hamiltonian Path: in the former some arcs are optional and other are compulsory, while in the latter all arcs are optional but vertices are compulsory. This constraint is easily implemented in OptionalRealizable$_2$ by duplicating each vertex and adding a compulsory arc between the two copies. Intuitively, this is the main building block of our reduction for OptionalRealizable$_w$ (Lemma 8 below). With $w \geq 3$, we further introduce $w-2$ *padding vertices* that will take place between any two consecutive vertices of the path, so that the window overlaps successive pairs of vertices along the path. These padding vertices also help enforce that no vertex is visited more than once.

Lemma 8. *For any fixed $w \geq 3$, OptionalRealizable$_w$ is NP-hard.*

Proof By reduction from Hamiltonian Path (see Lemma 7, 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 9 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^i_p for each $p \in [2n+1]$, $i \in [w-2]$. The overall vertex set is thus

$$V' = \{x^i_p \mid p \in [2n+1], i \in [w-2]\} \cup \bigcup_{v \in V} \{v_0, v_1\}$$

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

- arc (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 $v \in V$, $p \in [n]$, $i \in [w-2]$.
- arc (x^i_p, x^j_p) for $i < j$ and $p \in [2n+1]$;
- arc (x^i_p, x^j_{p+1}) for $j \leq i$ and $p \in [2n]$

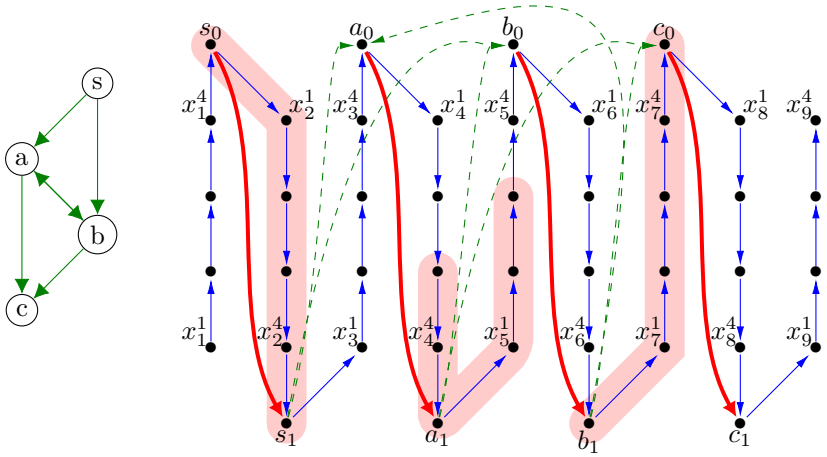


Fig. 9: Left: an instance G of Hamiltonian Path with a source vertex s and solution (s, a, b, c). Right: the corresponding instance G' of $\mathsf{OptionalRealizable}_6$. Heavy (red) arcs are compulsory, light (blue) arcs are a solution path in the graph, dashed (green) arcs are optional arcs issued 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.

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

$\Leftarrow$ Let $v^1, v^2, \ldots, v^n$ be an Hamiltonian path of G, so that v_0^p, v_1^p are the corresponding vertices in G', $p \in [n]$. Without loss of generality, since s has degree 1, $v^1 = s$. Define the following sequences:

$$X^p = x_{2p-1}^1 \ldots x_{2p-1}^{w-2} v_0^p x_{2p}^1 \ldots x_{2p}^{w-2} v_1^p \quad \text{for } p \in [n]$$
$$X^{n+1} = x_{2n}^1 \ldots x_{2n}^{w-2}$$
$$S = X^1 \ldots X^n X^{n+1}$$

We show that S is a solution for $\mathsf{OptionalRealizable}_w(G', P, A_c)$. By construction S starts with P. Further, for each compulsory arc (v_0, v_1), v is part of the Hamiltonian path so there is p such that $v = v^p$: 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 solution for $\mathsf{OptionalRealizable}_w(G', P, A_c)$. We first show that for each occurrence of x_p^i in S (except for $p = 2n+1$), the next $w-1$ characters in S are necessarily $x_p^{i+1} \dots x_p^{w-2} v_q x_{p+1}^1 \dots x_{p+1}^i$ for some v_q with $\in V$ and $q \in \{1, 2\}$. To this end, consider some size-w window xS' in S, where $x = x_p^i$ for some $i \in [w-2]$, $p \in [2n]$ (note that $p \neq 2n+1$). Let $T = x_p^{i+1} \dots x_p^{w-2}$ and $U = x_{p+1}^1 \dots x_{p+1}^i$ (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 pigeonhole principle string S' must contain at least one vertex v_q, $v \in V$. Since there are no arc (a_q, b_q) for $a, b \in V$, S' contains exactly one such 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 $x_p^1 \dots x_p^{w-2}$. From the arguments above, and the fact that S starts with X_1 (since P uses vertices of X_1), there exist vertices $u_1, \dots u_{2n}$ in $\bigcup_{v \in V} \{v_0, v_1\}$ such that

$$S = X_1 u_1 X_2 u_2 X_3 u_3 \dots u_{2n} X_{2n+1}$$

From the window size w, there must exist an arc (u_p, u_{p+1}) for each $p \in [2n-1]$. So if $u_p = v_0$ for some $p \in [2n-1]$ and $v \in V$, then $u_{p+1} = v_1$ (for the same v). Moreover, if $u_p = v_1$ for some $p \in [2n-1]$ and $v \in V$, then $u_{p+1} = v'_0$ for some vertex $v' \in V$ with $(v, v') \in A$. Overall vertices u_i alternate between $\{v_0 \mid v \in V\}$ and $\{v_1 \mid v \in V\}$, starting with $u_1 = s_0$ (by the starting sequence constraint), so there are vertices $(v^1, \dots, v^n)$ such that $u_{2p-1} = v_0^p$, $u_{2p} = v_1^p$, and $(v^1, \dots, v^n)$ is a walk of G.

Furthermore, arcs (v_0, v_1) are compulsory for each vertex $v \in V$, so $(v^1, \dots, v^n)$ must visit all n vertices of G: Thus, $(v^1, \dots, v^n)$ is an Hamiltonian path in G. □

We can now prove that $\mathsf{DU\text{-}Realizable}_w$ is NP-hard by reduction from $\mathsf{OptionalRealizable}_w$. The main idea of the reduction is to attach a gadget to the graph, that allows to visit each optional arc in some order; this becomes a prescribed prefix of any realization (denoted Z in the proof). Then, the end of a realization must visit all remaining (compulsory) arcs, but can still use any optional arc thanks to the unweighted setting. The gadget is heavily constrained to enforce that the prescribed prefix Z is indeed visited "as is" (in particular, without realizing any compulsory arc), and that vertices of the gadget are no longer accessible once Z is over.

Lemma 9. *For any fixed $w \geq 3$, $\mathsf{DU\text{-}Realizable}_w$ 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 $P = (s_1 \dots s_{w-1})$ of vertices of V. The following reduction is illustrated in Figure 10.

Let $m = |A_o|$, and write $A_o = \{(u_1, v_1), \dots, (u_m, v_m)\}$. Create G' by adding $w(m+1)+m$ *separator* vertices: $w(m+1)$ vertices y_p^i 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 following strings (using the product operator for concatenation)

$$Z = \left(\prod_{p=1}^{m} (y_p^1 \dots y_p^w u_p z_p v_p) \right) y_{m+1}^1 \dots y_{m+1}^w Z' \qquad = Z s_1 \dots s_{w-1}$$

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. In details, the additional arcs in G' are the following (where indices i, j, p necessarily satisfy $i \in [w]$, $j \in [w]$ and $p \in [m]$):

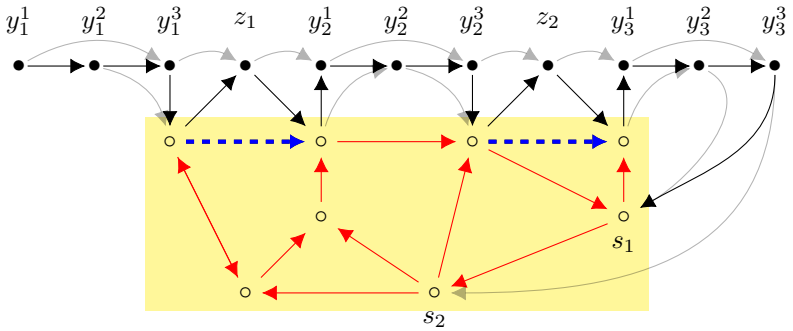


Fig. 10: Reduction from OptionalRealizable$_w$ to DU-Realizable$_w$ with $w = 3$. The input instance is the highlighted graph with white vertices (including two optional dashed blue arcs), as well as the starting sequence (s_1, s_2). The reduction adds the black vertices y_p^i and z_p, and the corresponding 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.

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

Claim 2. *G has a w-realization with optional arcs $\Leftrightarrow$ G' has a w-realization*

$\Rightarrow$ Build a realization for G' by concatenating Z with the realization for G starting with $s_1 \dots s_{w-1}$. 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 $Z[1, \dots, q]$ may only appear in $S[1, \dots, q]$.

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 y_1^1 does not appear again in S) 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 $N = \{Z'[q' + 1], \dots, Z'[q' + w - 1]\}$ without self-loops, so $S[q' + 1, \dots, q' + w - 1]$ is a

permutation of N. In case A, $w-2$ 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 $\mathsf{OptionalRealizable}_w$ with sequence S'.

□

4.2.2 Reduction for weighted problems

Now, let us prove that $\mathsf{GW\text{-}Realizable}_w$ and $\mathsf{DW\text{-}Realizable}_w$ are NP-hard for all $w \geq 3$, by reduction from the undirected variant of HP2 of Hamiltonian Path (cf Lemma 7). We focus on the directed case first, $\mathsf{DW\text{-}Realizable}_w$, the undirected case will simply use the underlying graph introduced in this reduction.

Lemma 10. *For any fixed $w \geq 3$, $\mathsf{DW\text{-}Realizable}_w$ and $\mathsf{GW\text{-}Realizable}_w$ are NP-hard.*

***Reduction for* DW-Realizable.**

Given $G = (V, E)$ undirected 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 11, as the union of the start gadget, the queue gadget, and the vertex and edge gadgets respectively for each vertex and edge of G. An example realization is given in Figure 12.

***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).

Correctness of the reduction.

We prove the following three claims:

Claim 3. *G Hamiltonian $\Rightarrow$ G' has a realization*

Claim 4. *G' has a realization $\Rightarrow$ G'_u has a realization*

Claim 5. *G'_u has a realization $\Rightarrow$ G is Hamiltonian*

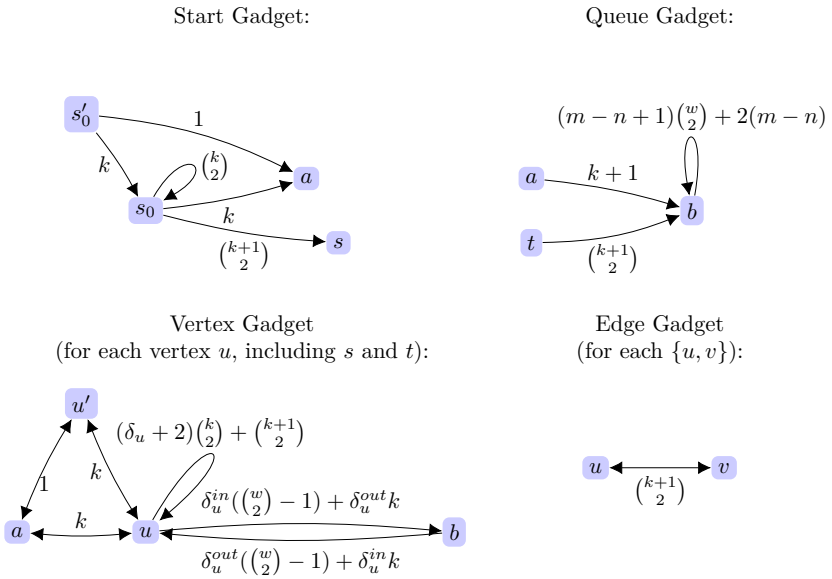


Fig. 11: Subgraphs used in the reduction from Hamiltonian Path to DW-Realizable$_3$. Weights on double arcs apply to both directions. Note that arcs (t, b) appear in two different gadgets, so their weights should be summed.

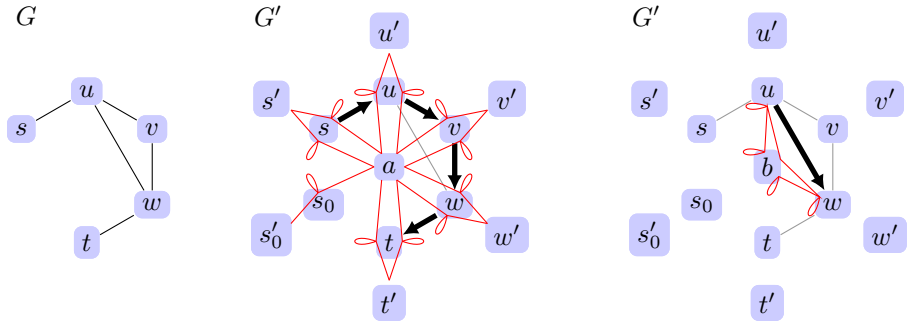


Fig. 12: Reduction from Hamiltonian Path to DW-Realizable. Left: the input graph G with degree-one vertices s and t (and Hamiltonian Path (s, u, v, w, t)). Each edge becomes two arcs (in each direction) in the edge gadgets of the resulting graph G'. Center: the first part of the path realizing most edges of G' (in particular those involving vertex a and primed versions of vertices), 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.

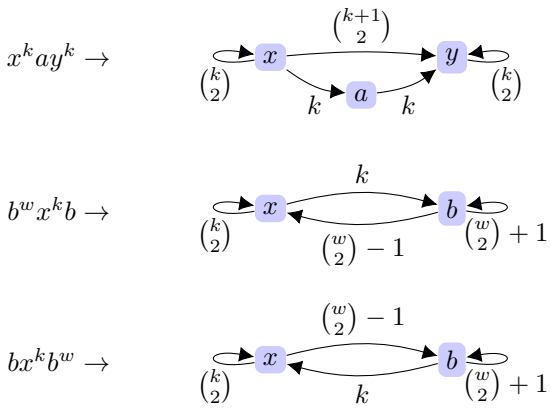


Fig. 13: Sequence graphs for strings of the form $x^k a y^k$, $b^w x^k b$ or $b x^k b^w$ for any x, y, a, b, with window size $w = k + 2$.

Proof of Claim 3 Assume that G has an 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 (recall that $k = w - 2$).

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

We verify for each gadget that all arcs are indeed realized with the correct weight[3], Figure 13 helps compute the weights of short string fragments. 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 uv (resp. vu) is part of the Hamiltonian path, then the arc (u, v) (resp. (v, u)) 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 4 Clearly, any realization for G' is a realization for G'_u. □

[3]For most arcs, the weights are not actually relevant and are only computed since they must be part of the input. Only weights of arcs incident to s'_0, a and vertices u' are used in the rest of the proof (proof of Claim 5). The central point here is that the sequence graph of S is the same for *any* sequence S obtained from an Hamiltonian path of G.

Proof of Claim 5 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 $k+1 = w-1$
- 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). Then 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 $au_i^* u'_i u_i^* a$. 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 an Hamiltonian path of G. □

All together, claims 3, 4 and 5 show the correctness of the reductions for both GW-Realizable$_a$nd DW-Realizable$_f$rom Hamiltonian Path (HP2) since they yield :

$$G \text{ is Hamiltonian } \Leftrightarrow G' \text{ has a realization}$$
$$G \text{ is Hamiltonian } \Leftrightarrow G'_u \text{ has a realization}$$

This completes the proof of Lemma 10.

4.3 Exponential algorithms for weighted problems

In spite of its NP-hardness, the weighted version of Realizable can be solved exactly for moderate instance sizes. In this section, we provide two complementary exponential-time algorithms, respectively based on Integer Linear Programming and Dynamic Programming, to effectively solve our problem in the context of tame instances. The latter runs in $\mathcal{O}(n^w\, 2^{wp})$, and produces the total number of realizations.

4.3.1 Linear integer programming formulation for DW- and GW-Realizable$_w$

Let $G = (V, E)$ be a graph with integer weights $\pi_{e \in E}$. We consider first the directed case DW, and show how our results extend to GW at the end of this section. In our model, we represent a sequence x over a size-n alphabet Vè, as a boolean matrix

$X \in \mathbb{M}_{n,p}(\{0,1\})$ encoding the sequence x:

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

The length-p sequences over V are thus in bijection with the boolean matrices such that $\forall j \in [p], \sum_{v \in V} X_{v,j} = 1$.

In order to encode sliding window constraints, we define the set $\mathcal{C}$ of all pairs of positions occurring together in a size-w window, $\mathcal{C} = \{(i,j) \mid i,j \in [p], i < j < i + w\}$. We use an intermediary slack variable $y^e_{i,j} \in \{0,1\}$ for each $(i,j) \in \mathcal{C}$ and $e = (v_1, v_2) \in E$ to model the appearance of v_1, v_2 together at indices i, j in a size-w window, i.e. $y^e_{i,j}$ is equal to 1 when v_1 is located at position j and v_2 at position $j + i$, and 0 otherwise. This is achieved with the following linear constraints

$$\begin{aligned} -X_{v_1,i} \qquad\qquad + y^e_{i,j} &\leq 0 \\ - X_{v_2,j} + y^e_{i,j} &\leq 0 \\ X_{v_1,i} + X_{v_2,j} - y^e_{i,j} &\leq 1 \end{aligned}$$

For a length-p sequence, the number of possible position pairs $(i,j) \in \mathcal{C}$ is given by:

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

We also need to forbid missing pairs not forming edges in the graph, which give the following constraint for each $(v_1, v_2) \notin E$ and every $(i,j) \in \mathcal{C}$

$$X_{v_1,i} + X_{v_2,j} \leq 1$$

It is worth noting that the value of p can be directly computed from the input, since the weight matrix and window size entirely constraints the realization size. More precisely, p is such that $|\mathcal{C}| = \sum_{e \in E} \pi_e$, which gives

$$p = \frac{w}{2} + \frac{\sum_{e \in E} \pi_e}{w - 1}$$

Then, $\mathsf{DW\text{-}Realizable}_w$ can be formulated as an integer linear program:

$$\min_{\substack{X \in \{0,1\}^{p \times n} \\ y \in \{0,1\}^{|E| \times |\mathcal{C}|}}} \sum_{e \in E} \sum_{i,j \in \mathcal{C}} y^e_{i,j}$$

$$\text{such that} \quad \forall j \in [p] \quad \sum_{v=1}^{n} X_{v,j} = 1$$

$$\begin{array}{ll} \forall e=(v_1,v_2)\in E & \begin{cases} -X_{v_1,i} & +y^e_{i,j}\leq 0 \\ -X_{v_2,j} & +y^e_{i,j}\leq 0 \\ X_{v_1,i}+X_{v_2,j} & -y^e_{i,j}\leq 1 \end{cases} \\ \forall (i,j)\in\mathcal{C} & \\ \forall e'=(v'_1,v'_2)\notin E & X_{v'_1,i}+X_{v'_2,j}\leq 1 \\ \forall (i,j)\in\mathcal{C} & \\ \text{and } \forall e\in E & \sum_{(i,j)\in\mathcal{C}} y^e_{i,j}\geq \pi_e \end{array}$$

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

For the undirected variant, for each edge $e=\{u,v\}$ with $u\neq v$ we build the ILP as if we had two directed edges (u,v) and (v,u). The only specificity is for the edge weight verification, that becomes the following:

$$\forall e=\{u,v\}\in E,\ \sum_{(i,j)\in\mathcal{C}} y^{(u,v)}_{i,j}+y^{(v,u)}_{i,j}\geq \pi_e.$$

This ensures that arcs (u,v) and (v,u) get a total weight of π_e, that can be shared in any way. We thus get an ILP for $\mathsf{GW\text{-}Realizable}_w$.

4.3.2 Dynamic programming algorithm for DW- and GW-NumRealizations$_w$

We present here a baseline dynamic programming algorithm which, despite having punishing complexity, allows to compute the number of realizations for modest instances of the weighted directed and undirected cases.

The recursion proceeds by extending a partial sequence, initially set to be empty, keeping track along the way of represented edges and of the vertices appearing in the last window. 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 $\Pi=(\pi_e)_{e\in E}$, preceded by a sequence of nodes $\mathbf{u}:=(u_1,\ldots,u_{|\mathbf{u}|})\in V^\star$. It can be shown that, for all $p\geq 1$, $\Pi\in\mathbb{N}^{|E|}$ and $\mathbf{u}\in V^{\leq w}$, $N_w[\Pi,p,\mathbf{u}]$ obeys the following formula in the directed case:

$$\begin{aligned} N_w[\Pi,p,\mathbf{u}] &= \sum_{v\in V}\begin{cases} N_w\left[\Pi'_{(\mathbf{u},v)},p-1,(u_1,...,u_{|u|},v)\right] & \text{if } |\mathbf{u}|<w-1 \\ N_w\left[\Pi'_{(\mathbf{u},v)},p-1,(u_2,...,u_{w-1},v)\right] & \text{if } |\mathbf{u}|=w-1 \end{cases} \\ \text{with } \Pi'_{(\mathbf{u},v)} &:= (\pi'_e)_{e\in E} \\ \pi'_e &:= \pi_e-|\{k\in[1,|\mathbf{u}|]\mid e=(u_k,v)\}| \end{aligned}$$

The base case of this recurrence corresponds to $p=0$, and is defined as

$$\forall\ \Pi,\ N_w[\Pi,0,\mathbf{u}]=\begin{cases}1 & \text{if } \Pi=(0)_{(i,j)\in V^2} \\ 0 & \text{otherwise.}\end{cases} \tag{5}$$

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.

A similar dynamic programming scheme holds in the undirected case, through a minor modification ($e = \{u_k, v\}$ in the definition of π'_e).

The overall recurrence for sequence length p can be computed in time $\mathcal{O}(|V|^w \times \prod_{e\in E}(\pi_e + 1))$ using memorization. This complexity can be refined by noting that:

$$\begin{aligned} \pi_e + 1 &\leq 2^{\pi_e} \\ \text{and } \sum_{e\in E} \pi_e &\leq w \times p \\ \text{so } \prod_{e\in E} (\pi_e + 1) &\leq \prod_{e\in E} 2^{\pi_e} \\ &\leq 2^{w\,p}. \end{aligned}$$

We thus obtain an overall complexity in $\mathcal{O}(n^w 2^{w\,p})$, improving on the trivial $\mathcal{O}(n^p)$ enumeration algorithm whenever $n > 2^w$. Then, 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. Precisely, 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 14 for an instance and the resulting sequences obtained by our algorithm.

5 Exponential lower bound on the size of realizations

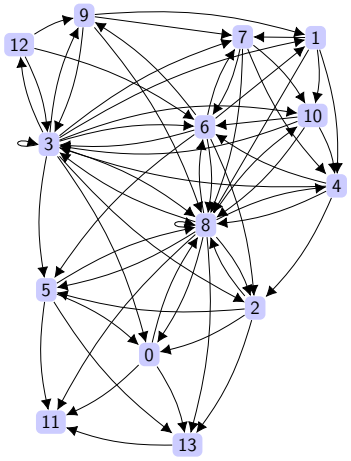
We have established in Section 4.2 the NP-hardness of several versions of realizability, yet have left open the question of their membership to NP. Such a property is usually proven by exhibiting a non-deterministic Turing machine which guesses a solution in polynomial time, and tests each of them in polynomial-time. This strategy requires that the (minimal) size of a solution grows only polynomially with the input length. Unfortunately, we find that the minimal size of a realization can grow exponentially larger that the input size, as formally stated below.

Proposition 6. *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 15 for an example. Our construction uses three sets of vertices A, B and C of size $k \times n$ each (vertices are labeled respectively $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ with $i \in [k]$ and $0 \leq j < n$), plus an additional *start* vertex s. A vertex $a_{i,j} \in A$ has *rank* i and *value* j. A vertex $b_{i,j} \in B$ or $c_{i,j} \in C$ has *rank* $i + k$ and *value* j. Vertex s has rank and value 0. Ranks are counted in $\mathbb{Z}/2k\mathbb{Z}$.

We consider k-tuples $T = (j_1, \ldots, j_k)$ with values in $[0, n-1]$. They are ordered according to the lexicographic order from $(0, \ldots, 0)$, $(0, \ldots, 1)$ to $(n-1, \ldots n-1)$. In particular, the successor of T is the k-tuple $T' = (j_1, \ldots, j_{x-1}, j_x + 1, 0, \ldots, 0)$ where x is the largest index such that $j_x < n-1$.

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



(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 with $w = 5$

Fig. 14: 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 $p = 20$.

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 $b_{i,j}$ 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, each $c_{i,j}$ with $1 \leq i \leq k$ and $0 \leq j < n$ has an outgoing arc to $c_{i',0}$ for $i' > i$, to each $a_{i',j'}$ with $i' < i$ and to $a_{i,j}$.

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 distinct ranks among $[r+1, r+k]$, which is only possible 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, $S[1, \dots, k+1] = sa_{0,0} \dots a_{k,0}$.

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 case where $S[p] = a_{i,j}$ and $S[p+k] = c_{i,j+1}$. Then by the outgoing arcs of $c_{i,j+1}$, necessarily $S[p+k+i'-i] = c_{i',0}$ for all $i < i' \leq k$. Let p be a position such that $S[p]$ has rank 1, let $T = (j_1, \dots, j_k)$ be the tuple of values of $S[p] \dots S[p+k-1]$, let T' be the tuple of values of $S[p+k] \dots S[p+2k-1]$, and T'' be the tuple of values of $S[p+2k] \dots S[p+2k-1]$. Then if $S[p+k] \dots S[p+2k-1]$ does not contain any vertex in C, then $T = T' = T''$. Otherwise, $T' = (j_1, \dots, j_{x-1}, (j_x + 1 \mod n), 0, \dots, 0)$ with x the smallest index such that $S[p+k+x] \in C$. In particular, $S[p+k+i'] = c_{i',0}$

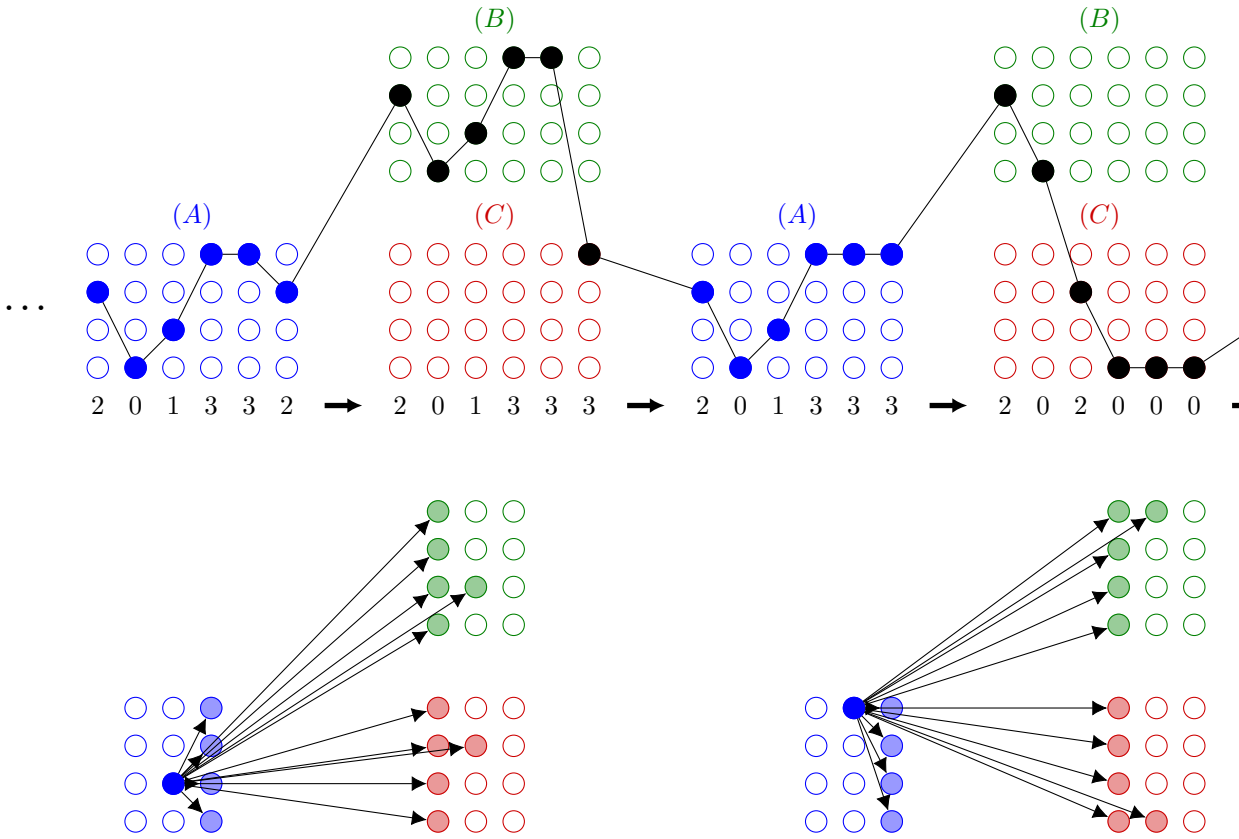


Fig. 15: Illustration of the construction in Proposition 6 for a graph with an exponentially long realization, with $n = 4$ and $k = 6$. Top: a fragment of the path, starting with the substring $a_{1,2}a_{2,0}a_{3,1}a_{4,3}a_{5,3}a_{6,2}$: in a correct realization, such a fragment (with value $(2, 0, 1, 3, 3, 2)$) 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.

and $S[p + i'] = a_{i',n-1}$ for each i' with $x < i' < k$. If $j_x = n - 1$, then T' is before T in lexicographic order (same prefix, and all values in an entire suffix is reset to 0). Otherwise $j_x < n - 1$ so x is the largest index such that $j_x < n - 1$ and T' is the successor of T

To conclude, S contains $a_{0,0} \dots a_{k,0}$, i.e. a substring with tuple of values $(0, \dots, 0)$. It also uses the arc $(a_{1,n-1}, c_{1,0})$, which implies that S also contains $a_{1,n-1} \dots a_{k,n-1}c_{1,0} \dots c_{k,0}$, hence S contains a substring with value tuple $(n - 1 \dots n - 1)$. For any two successive value tuples T, T' either T' is lower (or equal) to T in lexicographical order, either T' is the successor

of T, so overall S must use every possible tuple at least once (for substrings with ranks 1 to k). Thus S 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 $sa_{1,0} \dots a_{k,0}$ and ending with $a_{1,n-1} \dots a_{k,n-1}$), 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. □

6 Discussion

In this study, we have formalized a new series of inverse problems to assert the (un)ambiguity of popular word embeddings. We have provided a comprehensive characterization of their complexity, leaving only open their general membership in NP whenever $w \geq 3$. Indeed, given a sequence, computing its sequence graph representation can be done in $O(d^2 + p)$, if 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 proved that 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 6 for DU-Realizable.

Given the success of Large Language Models (LLMs), it would be of interest to consider similar inverse problems for LLM embeddings. In particular, this may lead to a better understand the ability of the LLMs to encode syntactic and semantic information. We think that the ones studied in this article would probably need to be adapted in order to gain some meaningful insights. Due to their over-parametrization, we suspect that the map between sequences and embeddings in that case is injective (for sequences up to a few thousand of symbols). In particular, NumRealizations would become equivalent to Realizable. More broadly, a challenging question of interest suggested by this study are the potential connections between the computational complexity of (well-chosen) inverse problems of embeddings (such as NumRealizations, Realizable, or variants thereof) and the capacity of the considered embedding to faithfully capture syntactic and semantic properties of natural language.

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

Author Contributions: All authors contributed equally to this work.

Funding: No funding to declare.

References

[1] Grohe, M.: word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pp. 1–16 (2020)

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[3] Phuong, M., Hutter, M.: Formal algorithms for transformers. arXiv preprint arXiv:2207.09238 (2022)

[4] Gordon, J., Lopez-Paz, D., Baroni, M., Bouchacourt, D.: Permutation equivariant models for compositional generalization in language. In: International Conference on Learning Representations (2019)

[5] Peyrard, M., Ghotra, S.S., Josifoski, M., Agarwal, V., Patra, B., Carignan, D., Kiciman, E., West, R.: Invariant language modeling. arXiv preprint arXiv:2110.08413 (2021)

[6] White, J.C., Cotterell, R.: Equivariant transduction through invariant alignment. arXiv preprint arXiv:2209.10926 (2022)

[7] Chomsky, N.: Syntactic Structures. Mouton de Gruyter, ??? (2002)

[8] Montague, R., et al.: Universal grammar. 1974, 222–46 (1970)

[9] Partee, B., *et al.*: Lexical semantics and compositionality. An invitation to cognitive science: Language **1**, 311–360 (1995)

[10] Petrache, M., Trivedi, S.: Position paper: Generalized grammar rules and structure-based generalization beyond classical equivariance for lexical tasks and transduction. arXiv preprint arXiv:2402.01629 (2024)

[11] 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)

[12] 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)

[13] 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)

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

[15] De Bruijn, N.G.: A combinatorial problem. Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam **49**(7), 758–764 (1946)

[16] Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. Siam Review **56**(1), 3–69 (2014)

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

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

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