Graphs can be succinctly indexed for pattern matching in $O(|E|^2 + |V|^{5/2})$ time

Nicola Cotumaccio
Gran Sasso Science Institute (GSSI)
L'Aquila, Italy
nicola.cotumaccio@gssi.it

Abstract
For the first time we provide a succinct pattern matching index for arbitrary graphs that can be built in polynomial time, while improving both space and query time bounds from [SODA 2021]. We show that, given an edge-labeled graph $G = (V,E)$, there exists a data structure of $|E/\leq G| ([\log |\Sigma|] + [\log q] + 2) \cdot (1 + o(1)) + |V/\leq G| \cdot (1 + o(1))$ bits which supports pattern matching on $G$ in $O(|P| \cdot q^2 \cdot \log(q \cdot |\Sigma|))$ time, where $G/\leq G = (V/\leq G, E/\leq G)$ is a quotient graph obtained by collapsing some nodes in $G$ and $q$ is the width of the maximum co-lex relation on $G$. Our results have relevant applications in automata theory: we can use our data structure to decide whether a string belongs to the language recognized by a given automaton, and we can capture the degree of nondeterminism of an NFA.

1 Introduction
Pattern matching is a pervasive problem in computer science: given a pattern and some data, decide whether the pattern matches the data. In this paper, we will consider graph pattern matching.

Graph pattern matching Let $\Sigma$ be an alphabet, and let $G = (V,E)$ be an edge-labeled graph. The pattern-matching problem is the following: given a pattern $P \in \Sigma^*$, decide whether $\alpha$ can be read on $G$ by following edges whose labels, when concatenated, yield $P$.

The problem of matching patterns on graphs arises in a number of fields. In bioinformatics, the pan-genome is a labeled graph capturing the genetic variation within a species [1,2]. Pattern matching on labeled graphs is also natural in graph databases [3,4].

In this paper we consider the problem of building an index for pattern matching. In other words, we aim to preprocess a given graph in such a way that we can quicker answer multiple pattern matching queries. At the same time, we want to employ a succinct data structure [5], that is, a data structure that requires a number of bits close to the lower bounds from information theory, while allowing efficient queries.

The problem of indexing node-labeled graphs for pattern matching has been extensively studied in the last years. When searching for a match on node-labeled graphs, one still follows edges, and a string is read by concatenating the labels on the nodes. The problem of pattern matching on edge-labeled graphs is more general, because a node-labeled graph can be thought of as an edge labeled graph where all edges entering the same node have the same label.
On the one hand, in [6] Equi et al. showed that there exists no algorithm indexing a node-labeled graph $G = (V,E)$ in polynomial time in such a way that a pattern $P \in \Sigma^*$ can be checked for pattern matching in $O(|E|^\delta |P|^\beta)$ time, where $\delta < 1$ or $\beta < 1$, unless the Orthogonal Vector Hypothesis (OVH) is false. On the other hand, in [7] Gagie et al. introduced a class of graphs - the so-called Wheeler graphs - which can be succinctly stored while allowing pattern matching in $O(|P| \log |\Sigma|)$ time. Wheeler graphs generalize a number of previous approaches based on sorting the nodes of the graph. Such graphs are endowed with a total order (a Wheeler order) that satisfies path coherence: if one starts from an interval of nodes and follow all edges labeled with a letter $a$, then one still ends up in an interval of nodes. Note that the bound $O(|P| \log |\Sigma|)$ does not break the bound $O(|E|^\delta |P|^\beta)$ because most graphs are not Wheeler graphs. For example, it can be showed that a unary language is recognized by some Wheeler automaton (that is, an automaton whose underlying graph is Wheeler) if and only if the language is finite or cofinite [8]. Another limitation of Wheeler graphs is that all natural problems connected with the property of being Wheeler are hard: in, particular, deciding whether a node-labeled graph is Wheeler is an NP-complete problem [9].

In [10] the indexing techniques for Wheeler graphs were extended to arbitrary node-labeled graphs. The main idea is to consider orders on the set of nodes that are allowed to be partial, the so-called co-lex orders. In general, if $(V, \leq)$ is a partial order, we can consider a partition $\{V_i\}_{i=1}^p$ of $V$ such that, for every $i$, every pair of elements in $V_i$ are $\leq$-comparable. The minimum size of such a partition is the width of the partial order. It can be showed that a co-lex order of width $p$ can be used to succinctly index the graph in a such a way that pattern matching queries can be solved in $O(|P| \cdot p^2 \cdot \log(p \cdot |\Sigma|))$ time. In particular, a co-lex order is a Wheeler order if and only if $p = 1$, and in this case we retrieve the bound $O(|P| \log |\Sigma|)$. While most node-labeled graphs do not admit a Wheeler order, every node-labeled graph admits a co-lex order. Both the bound $O(|P| \cdot p^2 \cdot \log(p \cdot |\Sigma|))$ and the (succinct) number of bits required to index the graph are proportional to $p$, so one should determine a co-lex order of width as small as possible. However, co-lex orders inherit the hardness of the problems connected to Wheeler orders: determining the minimum width of a co-lex order on a graph is NP-hard (we mentioned that the simpler problem of determining whether a graph is Wheeler is already NP-complete). This implies that the problem of indexing a graph with the best (i.e., minimum-width) co-lex order is hard. In addition, no approximation algorithm for computing such a minimum width (and the corresponding co-lex order) is currently known.

2 Our contribution

Wheeler orders and co-lex orders allow efficient pattern matching because they ensure (some variant of) path coherence. Since sorting the nodes of graphs leads to place restrictions - such as Wheelerness - being NP-hard to check, it is natural to wonder whether it is in fact necessary to rely on sorting. Defining a total order implies ensuring antisymmetry and transitivity, and these properties are logically hard to express. For example, in [11] it was showed that on a special class of graphs (2-NFAs) the problem of deciding whether a given graph is Wheeler can be solved in polynomial time, and the main idea is to reduce the problem to 2-SAT by defining clauses expressing the property of being Wheeler. The reason
why this method does not work for arbitrary graphs is that one should in particular define a clause for expressing transitivity, and transitivity requires 3-SAT clauses on general graphs. Similarly, antisymmetry is not a necessary constraint from a pattern matching perspective, because we will show that if two nodes are comparable in both directions, then they can be essentially thought of as a unique node.

To sum up, in this work we change perspective switching from (partial) orders to arbitrary relations, so defining co-lex relations. By removing antisymmetry and transitivity, we show that the notion of path coherence still makes perfect sense, and the algebraic structure behind pattern matching becomes cleaner. Indeed, we show that every graph \( G \) admits a maximum co-lex relation, that is, a co-lex relation \( R \) such that every co-lex relation on \( G \) is a restriction of \( R \) (while in general a graph does not admit a maximum co-lex order). In particular, the width of the maximum co-lex relation is automatically the minimum width of a co-lex relation on the graph. Moreover (1) the maximum co-lex relation can be computed in \( O(|E|^2) \) time and (2) it is always transitive. While transitivity is not conceptually relevant for path coherence, from an algorithmic perspective it is helpful for indexing. Moreover, we show that all nodes being comparable in both directions can be compressed into a single node. More precisely, we show that starting from a graph \( G \) one can always build a quotient graph \( G/\leq_{\alpha} \) that captures exactly the same information for pattern matching: one can always answer a query on \( G \) by answering the same query on \( G/\leq_{\alpha} \). This approach is successful because the graph \( G/\leq_{\alpha} \) is topologically simpler than the original graph \( G \): if a node in \( G/\leq_{\alpha} \) has been obtained by collapsing two or more nodes of the original graph, than such a node can have at most one incoming edge in the quotient graph. Moreover, \( G/\leq_{\alpha} \) always admits a maximum co-lex order (while a general graph does not admit a maximum co-lex order, as stated above), which is naturally induced by the maximum co-lex relation on \( G \). Since \( G/\leq_{\alpha} \) admits the maximum co-lex order and, crucially, it can be built in polynomial time, we can index \( G \) by simply indexing \( G/\leq_{\alpha} \) using the techniques from [10].

Let us state our quantitative results. In [10] it was showed that a node-labeled graphs \( G = (V,E) \) can be succinctly indexed by means of a data structure of \( |E|(|\log |\Sigma|| + \log p| + 2) \cdot (1 + o(1)) + |V| \cdot (1 + o(1)) \) bits which supports pattern matching in \( O(|P| \cdot p^2 \cdot \log(p \cdot |\Sigma|)) \) time, where \( p \) is the minimum width of a co-lex order on \( G \), \( P \) is the pattern and \( \Sigma \) is the alphabet. Determining \( p \) is NP-hard and building the data structure is also hard. In this paper, we work in the more general setting of edge-labeled graphs \( G = (V,E) \), and for the first time we provide a succinct index for arbitrary graphs that can be built in polynomial time (while even only determining if a graph is Wheeler is NP-hard), which requires less space and answers queries more efficiently than the one in [10]. We show that, given an edge-labeled graph \( G = (V,E) \), there exists a data structure of \( |E/\leq_{\alpha}|(|\log |\Sigma|| + \log q| + 2) \cdot (1 + o(1)) + |V/\leq_{\alpha}| \cdot (1 + o(1)) \) bits which supports pattern matching on \( G \) in \( O(|P| \cdot q^2 \cdot \log(q \cdot |\Sigma|)) \) time, where \( G/\leq_{\alpha} = (V/\leq_{\alpha}, E/\leq_{\alpha}) \) is the quotient graph (so \( |V/\leq_{\alpha}| \leq |V| \) and \( |E/\leq_{\alpha}| \leq |E| \)) and \( q \) is the width of the maximum co-lex relation on \( G \). The bounds achieved in this paper look similar to the ones in [10], but, in fact, there are several sources of improvement:

1. Most importantly, \( q \) can be determined in \( O(|E|^2) \) time, and our data structure can be built in \( O(|E|^2 + |V/\leq_{\alpha}|^{5/2}) \) time (while determining \( p \) and building the data structure in [10] are hard problems).

2. It always holds that \( q \leq p \), and \( q \) can be arbitrarily smaller than \( p \) (that is, for every
3. Our bound only depends on the size of $G / \leq_G$ and it is independent of the size of $G$. In other words, $G / \leq_G$ eliminates the unnecessary redundancy for pattern matching.

Due to space constraints, most proofs are omitted. In the extended version of this paper [12] we also present both theoretical implications and algorithmic applications to automata theory.

3 Notation

Let $\Sigma$ be an alphabet, and let $\leq$ be a fixed, total order on $\Sigma$. We denote by $G = (V, E)$ an (edge-labeled) graph, where $V$ is the set of nodes, and $E \subseteq V \times V \times \Sigma$ is the set of labeled edges. In this papers, all graphs are finite.

If $V$ is a set, a (binary) relation $R$ on $V$ is a subset of $V \times V$. We say that $u, v \in V$ are $R$-comparable if $(u, v) \in R \vee (v, u) \in R$ (note that $(u, v) \in R$ and $(v, u) \in R$ may be both true). If $R$ and $R'$ are binary relations on $V$, we say that $R$ refines $R'$ if $(u, v) \in R \implies (u, v) \in R$.

For $R$ a binary relation on $V$ and $U \subseteq V$, we say that $U$ is $R$-convex if:

$$(\forall u, v, z \in V)((u, z \in U \land (u, v) \in R \land (v, z) \in R) \implies v \in U).$$

A preorder $\leq$ on $V$ is a binary relation being reflexive and transitive. We write $u < v$ if $u \leq v$ and $u \neq v$. Moreover, the preorder $\leq$ is a partial order if it antisymmetric, and it is a total order if it is a partial order and every pair of elements are $\leq$-comparable.

We introduce some notation typical of partial order, and we naturally extend it to preorders. Let $(V, \leq)$ be a preorder. A set $V' \subseteq V$ is a $\leq$-chain if every $u, v \in V$ are $\leq$-comparable. A partition $\{V_i\}_{i=1}^{p}$ of $V$ is a $\leq$-chain partition if every $V_i$ is a $\leq$-chain. The width of $(V, \leq)$ is the minimum size of a $\leq$-chain partition. Note that if $\leq$ and $\leq'$ are preorders on $V$, and $\leq$ refines $\leq'$, then the width of $\leq$ is smaller than or equal to the width of $\leq'$ (because every $\leq'$-chain partition is also a $\leq$-chain partition).

Let us recall a standard method for obtaining a partially-ordered quotient set from a preorder. Let $(V, \leq)$ be a preorder. For every $u, v \in V$, let $u \sim \leq v$ if and only if $(u \leq v) \land (v \leq u)$. It is immediate to check that $\sim$ is an equivalence relation. Now, let $[v]_{\leq}$ be the quotient class of $v$, and consider the quotient set $V_{/\leq} = \{[v]_{\leq} | v \in V\}$. Define $\leq_{/\sim}$ on $V_{/\leq}$ by letting $[u]_{\leq} \leq_{/\sim} [v]_{\leq}$ if and only if $u \leq v$. The definition of $\sim_{\leq}$ implies that $\leq_{/\sim}$ is well-defined (that is, the definition does not depend on the choice of representatives), because if $u \sim_{\leq} u', v \sim_{\leq} v'$ and $u \leq v$, then $u' \leq u \leq v \leq v'$. Moreover $(V_{/\sim}, \leq_{/\sim})$ is a partial order. Indeed, if $[u]_{\sim} \leq_{/\sim} [v]_{\leq}$ and $[v]_{\leq} \leq_{/\sim} [u]_{\leq}$, then $u \leq v$ and $v \leq u$, so $[u]_{\leq} = [v]_{\leq}$.

4 Definitions and first results

Let $G = (V, E)$ be a graph. Let $\# \not\in \Sigma$ be a special symbol, and assume $\# < a$ for all $a \in \Sigma$. For $v \in V$ define:

$$\lambda(v) = \begin{cases} \{a \in \Sigma | (u, v, a) \in E \text{ for some } u \in V\} & \text{if } v \text{ has incoming edges} \\ \{\#\} & \text{if } v \text{ does not have incoming edges.} \end{cases}$$

275
In a Wheeler order, all nodes without incoming edges must come before all remaining nodes [7]. Intuitively, we let \( # < a \) for all \( a \in \Sigma \) to ensure a similar properties for arbitrary relations.

If \( u, v \in Q \), define:

\[
\lambda(u) \angle \lambda(v) \iff (\forall a \in \lambda(u))(\forall b \in \lambda(v))(a \preceq b).
\]

We can now give our main definition, which generalizes the definition of co-lex order given in [10] (the term "co-lex order" refers to the co-lexicographic ordering of strings induced by a co-lex order, see the extended version [12]).

**Definition 1.** Let \( G = (V, E) \) be a graph. A co-lex relation on \( G \) is a reflexive relation \( R \subseteq V \times V \) that satisfies the following two axioms:

1. **(Axiom 1)** For every \( u, v \in V \) such that \( u \neq v \), if \( (u, v) \in R \), then \( \lambda(u) \angle \lambda(v) \);
2. **(Axiom 2)** For every \( (u', u, a), (v', v, a) \in E \) such that \( u \neq v \), if \( (u, v) \in R \), then \( (u', v') \in R \).

A co-lex preorder is a co-lex relation that is also a preorder. A co-lex order is a co-lex relation that is also a partial order.

**Remark 2.** Every graph \( G = (V, E) \) admits a co-lex relation. For example, \( \{(v, v) \mid v \in V\} \) and \( \{(u, v) \in V \times V \mid (\forall a \in \lambda(u))(\forall b \in \lambda(v))(a \prec b)\} \cup \{(v, v) \mid v \in V\} \) are co-lex relations on \( G \).

The property that allows to index a Wheeler graph for pattern matching is path coherence: starting from an interval of nodes and reading a string one still ends up in an interval of nodes [7]. This property was generalized to co-lex orders [10], and we now generalize it to arbitrary co-lex relations.

**Lemma 3** (Path coherence). Let \( G = (V, E) \) be a graph, and let \( R \) be a co-lex relation on \( G \). Let \( \alpha \in \Sigma^* \), and let \( U \subseteq V \) be \( R \)-convex. Then, the set \( U' \) of all nodes in \( V \) that can be reached from \( U \) by following edges whose labels, when concatenated, yield \( \alpha \), is still \( R \)-convex (possibly \( U' \) is empty).

Let us introduce the maximum co-lex relation.

**Definition 4.** Let \( G = (V, E) \) be a graph. Let \( R \) be a co-lex relation on \( G \). We say that \( R \) is maximum if it refines every co-lex relation \( R' \) on \( G \).

It is clear that if a maximum co-lex relation exists, then it is unique. The following lemma shows that the maximum co-lex relation always exists. This is a crucial distinction between co-lex relations and co-lex orders: in general, the maximum co-lex order - that is, a co-lex order refining every co-lex order - does not exist (see Figure 1), and this provides some intuition about why determining the minimum width \( p \) of a co-lex order on a graph is NP-hard.

**Lemma 5.** Every graph \( G = (V, E) \) admits the maximum co-lex relation (in the following denoted by \( \leq_G \)). Moreover, \( \leq_G \) is a co-lex preorder.
It is easy to check that the union of two co-lex relations is still a co-lex relation; however, in general the union of two co-lex orders is not a co-lex order, see Figure 1, so the maximum co-lex relation $\leq_G$ is the union of all co-lex relations on $G$. Moreover, $\leq_G$ is transitive because the transitive closure of a co-lex relation is still a co-lex relation.

If the maximum co-lex relation is also antisymmetric, then it also the maximum co-lex order; however in general the maximum co-lex order does not exist, or if it exists it can be distinct from the maximum co-lex relation (and in this case the maximum co-lex relation is a strict refinement of the maximum co-lex order), see Figure 1.

We now show that the maximum co-lex relation can be computed in $O(|E|^2)$ time. To this end, we need the characterization in Lemma 7. Since the maximum co-lex relation is transitive, when indexing a graph we can assume that we use a co-lex preorder.

**Definition 6.** Let $G = (V,E)$ be a graph, and let $(u',v'), (u,v) \in V \times V$ be pairs of distinct nodes. We say that $(u',v')$ precedes $(u,v)$ if there exist $u_1, \ldots, u_r, v_1, \ldots, v_r \in V$ ($r \geq 1$) and $a_1, \ldots, a_{r-1} \in \Sigma$ such that (1) $u_1 = u'$ and $v_1 = v'$, (2) $u_r = u$ and $v_r = v$, (3) $u_i \neq v_i$ for $i = 1, \ldots, r$ and (4) $(u_i, u_{i+1}, a_i), (v_i, v_{i+1}, a_i) \in E$ for $i = 1, \ldots, r-1$.

**Lemma 7.** Let $G = (V,E)$ be a graph, and let $u, v \in V$ be distinct nodes. Then:

$$u \prec_G v \iff \text{for all pairs } (u',v') \text{ preceding } (u,v) \text{ it holds } \lambda(u') \not\prec \lambda(v').$$

**Theorem 8.** Let $G = (V,E)$ be a graph. Then, $\leq_G$ can be computed in $O(|E|^2)$ time.

**Proof.** Consider the graph $G = (V, \mathcal{E})$, where $V = \{(u,v) \in V \times V | u \neq v\}$ and $\mathcal{E} = \{(u',v'), (u,v) \in V | (u',u,a), (v',v,a) \in E \text{ for some } a \in \Sigma\}$. First, mark all $(u,v) \in V$ such that $\lambda(u) \not\prec \lambda(v)$ does not hold true (the property ”$\lambda(u) \not\prec \lambda(v)$” can be checked in constant time because one only needs to compare the largest element in $\lambda(u)$ and the smallest element in $\lambda(v)$). Then, mark all nodes in $V$ reachable by a marked node. Notice that at the end a pair $(u,v)$ is marked if and only if there exists a pair $(u',v')$ preceding $(u,v)$ for which $\lambda(u') \not\prec \lambda(v')$ does not hold true, if and only if it holds $u \not\prec_G v$ (by Lemma 7). As a consequence, $\leq_G$ is the reflexive closure of the relation consisting of all non-marked nodes in $V$. Notice that $\leq_G$ can be computed in $O(|E|^2)$ time because $|\mathcal{E}| \leq |E|^2$ and nodes in $V$ can be marked by means of a graph traversal. \hfill $\square$
5 Quotienting a preorder

As stated in Section 2, we aim to build a quotient graph that captures all information required for pattern matching. Broadly speaking, we will construct the quotient graph starting from a co-lex preorder $\leq$ on $V$ and considering the partial order $(V/\leq, \leq\sim)$. In this section, we present some preliminary results that will be useful in the following.

**Lemma 9.** Let $(V, \leq)$ be a preorder. Then, the width of the partial order $(V/\leq, \leq\sim)$ is equal to the width of $(V, \leq)$.

Let us prove a simple result relating convexity and quotients: every convex set is the union of some $\sim\leq$-classes. This result is crucial for showing that without loss of generality we can perform pattern matching on the quotient graph.

**Lemma 10.** Let $(V, \leq)$ be a preorder, and let $U \subseteq V$ be $\leq$-convex. If $v \in U$, then $[v] \subseteq U$.

In other words, every $\leq$-convex set is the union of some $\sim\leq$-classes.

More generally, we can prove that there is a natural 1-1 correspondence between $\leq$-convex sets in $V$ and $\sim\leq$-convex sets in $V/\leq$.

**Lemma 11** (Correspondence theorem - convex sets). Let $(V, \leq)$ be a preorder. Let $U$ be the family of all $\leq$-convex sets in $V$, and let $U_\leq$ be the family of all $\leq\sim$-convex sets in $V/\leq$.

Define:

$$\phi : U \to U_\leq$$

$$\psi : U_\leq \to U$$

$$U \mapsto \{[v]_\leq | v \in U\}$$

$$U_\leq \mapsto \{v \in V | [v]_\leq \in U_\leq\}.$$

Then, $\phi$ is a bijective function, with inverse $\psi$.

6 The quotient graph

We can now define our quotient graph.

**Definition 12.** Let $G = (V, E)$ be a graph, and let $\leq$ be a co-lex preorder on $G$. Define $G/\leq = (V/\leq, E/\leq)$ by:

1. $V/\leq = \{[v]_\leq | v \in V\}$;
2. $E/\leq = \{(\{u\}_\leq, [v]_\leq, a) | (u', v', a) \in E \text{ for some } u' \in [u]_\leq \text{ and } v' \in [v]_\leq\}$.

Let us prove that $G/\leq$ enjoys a number of properties. (1) If a node of $G/\leq$ has been obtained by collapsing two or more nodes of $G$, then that node has at most one ingoing edge in $G/\leq$ (which is possibly a self-loop). (2) $\leq\sim$ is a co-lex order on $G/\leq$. (3) The graph $G/\leq G$ always admits the maximum co-lex order (recall that in general a graph does not admit the maximum co-lex order). More precisely, the maximum co-lex order is $\leq\sim G$ (the partial order on $V/\leq G$ induced by $\leq G$), which is also the maximum co-lex relation on $G/\leq G$. Notice that $G/\leq G$ is well-defined because $\leq G$ is a co-lex preorder by Lemma 5.

**Lemma 13.** Let $G = (V, E)$ be a graph, and let $\leq$ be a co-lex preorder on $G$. If $[v]_\leq \in V/\leq$ is such that $|[v]_\leq| \geq 2$, then there exists at most one edge entering $[v]_\leq$ in $G/\leq$.
Lemma 14. Let $G = (V, E)$ be a graph, and let $\leq$ be a co-lex preorder on $G$. Then, $\leq^\sim$ is a co-lex order on $G/\leq$, and the width of $\leq^\sim$ is equal to the width of $\leq$.

Lemma 15 (Correspondence theorem - co-lex relations). Let $G = (V, E)$ be a graph, and let $\leq$ be a co-lex preorder on $G$. Let $C_\leq$ the set of all co-lex relations on $G/\leq$, and let $C$ be set of all co-lex relations $R$ on $G$ such that, if $(u, v) \in R$, $[u]_\leq = [u]'_\leq$ and $[v]_\leq = [v]'_\leq$, then $(u', v') \in R$. Define:

$$
\rho : C_\leq \to C \\
R^\leq \mapsto \{(u, v) \in V \times V | ([u]_\leq, [v]_\leq) \in R^\leq \} \\
\sigma : C \to C_\leq \\
R \mapsto \{([u]_\leq, [v]_\leq) \in V/\leq \times V/\leq | (u, v) \in R\}.
$$

Then, $\rho$ is a bijective function, with inverse $\sigma$. In particular, $\sigma(\leq)$ is equal to $\leq^\sim$.

Corollary 16. Let $G = (V, E)$ be a graph. Then, $\leq^\sim_G$ is the maximum co-lex relation and the maximum co-lex order on $G/\leq_G$.

7 Indexing for pattern matching

Recall that in [10] it was showed how to index a graph by means of a co-lex order. However, determining a co-lex order of minimum width is a hard problem [10]. On the other hand, Corollary 16 ensures that $G/\leq_G$ always admits the maximum co-lex order (which has minimum width), and it can be determined in polynomial time by Theorem 8. As a consequence, we have overcome the hardness of determining a co-lex order of minimum width of an arbitrary graph if we show that we can answer pattern matching queries on $G$ by answering a query on $G/\leq_G$. This in indeed the purpose of the following lemma. Intuitively, if we start from a $\leq_G$-convex set $U$ of nodes in $G$, we can obtain the $\leq_G$-convex set of nodes that can be reached through a string $\alpha$ by (1) passing to the quotient, (2) obtaining the $\leq^\sim_G$-convex set of nodes that can reached through $\alpha$ in $G/\leq_G$, and (3) going back to $G$.

Lemma 17 (Correspondence theorem - path coherence). Let $G = (V, E)$ be a graph, and let $\leq$ be a co-lex preorder on $G$. Let $\alpha \in \Sigma^*$. Let $U$ be the family of all $\leq$-convex sets in $V$, and let $U_\leq$ be the family of all $\leq^\sim$-convex sets in $V/\leq$. Let:

$$
\theta_\alpha : U \to U \\
\theta^\leq_\alpha : U_\leq \to U_\leq
$$

be the functions such that if $U \in U$, then $\theta_\alpha(U)$ is the set of all nodes of $G$ that can be reached from $U$ by following edges whose labels, when concatenated, yield $\alpha$, and if $U_\leq \in U_\leq$, then $\theta^\leq_\alpha(U_\leq)$ is the set of all nodes of $G/\leq$ that can be reached from $U_\leq$ by following edges whose labels, when concatenated, yield $\alpha$. Let $\phi$ and $\psi$ the functions defined in Lemma 11. Then (see Figure 2):

$$
\theta_\alpha \circ \psi = \psi \circ \theta^\leq_\alpha \\
\phi \circ \theta_\alpha = \theta^\leq_\alpha \circ \phi.
$$
Corollary 18. Let $G = (V, E)$ be a graph, and let $\preceq$ be a co-lex preorder on $G$. Let $\alpha \in \Sigma^*$. Then the pattern matching problem returns "yes" on input $G$ and $\alpha$ if and only if it returns "yes" on input $G/\preceq$ and $\alpha$.

We now recall the main result from [10] and we adapt it to edge-labeled graphs.

**Theorem 19.** Let $G = (V, E)$ be a graph, and assume that we are given a co-lex order $\preceq$ on $G$ of minimum width $p$. Then, there exists a data structure of $|E|(|\log |\Sigma|| + \log p) + 2 \cdot (1 + o(1)) + |V| \cdot (1 + o(1))$ bits, which can be built starting from $G$ and $\preceq$ in $O(|V|^{3/2})$ time, that solves the pattern-matching problem in $O(|P| \cdot p^2 \cdot \log(p \cdot |\Sigma|))$ time, where $P \in \Sigma^*$ is the pattern.

Theorem 19 simply generalizes the result in [10] from node-labeled graphs to edge-labeled graphs. However, building the data structure in Theorem 19 does not require only $O(|V|^{3/2})$ time, because the theorem assumes that we are given a co-lex order $\preceq$ on $G$ of minimum width $p$, and determining such a co-lex order is a hard problem [10].

We can now overcome this limitation by passing to the quotient graph. Here is our main result.

**Theorem 20.** Let $G = (V, E)$ be a graph, and let $q$ be the the width of $\preceq_G$. Then, there exists a data structure of $|E/\preceq_G|(|\log |\Sigma|| + \log q) + 2 \cdot (1 + o(1)) + |V/\preceq_G| \cdot (1 + o(1))$ bits, which can be built starting from $G$ in $O(|E|^2 + |V/\preceq_G|^5/2)$ time, that solves the pattern-matching problem in $O(|P| \cdot q^2 \cdot \log(q \cdot |\Sigma|))$ time, where $P \in \Sigma^*$ is the pattern.

**Proof.** Compute $\preceq_G$ in $O(|E|^2)$ time (Theorem 8). Build the graph $G/\preceq_G$ by a graph traversal. By Lemma 14 we know that $\preceq_G$ is a co-lex order on $G/\preceq_G$ of width $q$. Moreover, $\preceq_G$ is the maximum co-lex order on $G/\preceq_G$ by Corollary 16, and so it is a co-lex order of minimum width. Hence, just build the data structure from Theorem 19 starting from $G/\preceq_G$ and $\preceq_G$. Corollary 18 ensures that querying $G/\preceq_G$ is equivalent to querying $G$.

The parameter $q$ in Theorem 20 is always smaller than or equal to the parameter $p$ in Theorem 19, because the maximum co-lex relation on $G$ refines every co-lex order on $G$. Moreover, $q$ can be arbitrarily smaller than $p$: for every integer $n$ there exists a graph for which $q = 1$ and $p = n$ (see Figure 3).
8 References

[1] Uwe Baier, Timo Beller, and Enno Ohlebusch, “Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform,” *Bioinformatics*, vol. 32, no. 4, pp. 497–504, 2015.

[2] Jouni Sirén, Niko Välimäki, and Veli Mäkinen, “Indexing graphs for path queries with applications in genome research,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 2, pp. 375–388, 2014.

[3] David W. Williams, Jun Huan, and Wei Wang, “Graph database indexing using structured graph decomposition,” in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 976–985.

[4] Renzo Angles and Claudio Gutierrez, “Survey of graph database models,” *ACM Comput. Surv.*, vol. 40, no. 1, Feb. 2008.

[5] Gonzalo Navarro, *Compact Data Structures: A Practical Approach*, Cambridge University Press, 2016.

[6] Massimo Equi, Veli Mäkinen, and Alexandru I. Tomescu, “Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless seth fails,” in *SOFSEM 2021: Theory and Practice of Computer Science*, Tomáš Bureš, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdziński, Claus Pahl, Florian Sikora, and Prudence W.H. Wong, Eds., Cham, 2021, pp. 608–622, Springer International Publishing.

[7] Travis Gagie, Giovanni Manzini, and Jouni Sirén, “Wheeler graphs: A framework for BWT-based data structures,” *Theoretical Computer Science*, vol. 698, pp. 67–78, 2017, Algorithms, Strings and Theoretical Approaches in the Big Data Era (In Honor of the 60th Birthday of Professor Raffaele Giancarlo).

[8] Jarno Alanko, Giovanna D’Agostino, Alberto Policriti, and Nicola Prezza, “Wheeler languages,” *CoRR*, vol. abs/2002.10303, 2020.

[9] Daniel Gibney and Sharma V. Thankachan, “On the hardness and inapproximability of recognizing wheeler graphs,” in *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, G. Herman M. A. Bender, O. Svensson, Ed. 2019, vol. 144 of LIPIcs, pp. 51:1–51:16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[10] Nicola Cotumaccio and Nicola Prezza, “On indexing and compressing finite automata,” in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2585–2599.

[11] Jarno Alanko, Giovanna D’Agostino, Alberto Policriti, and Nicola Prezza, “Regular languages meet prefix sorting,” in *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 911–930.

[12] Nicola Cotumaccio, “Graphs can be succinctly indexed for pattern matching in \(O(|E|^2 + |V|^{5/2})\) time,” *CoRR*, vol. abs/2111.04595, 2021.