How to transform graph states using single-qubit operations:
computational complexity and algorithms

Axel Dahlberg Jonas Helsen Stephanie Wehner

May 16, 2018

Abstract

Graph states are ubiquitous in quantum information with diverse applications ranging from quantum network protocols to measurement based quantum computing. Here we consider the question whether one graph (*source*) state can be transformed into another graph (*target*) state, using a specific set of quantum operations (LC + LPM + CC): single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC) between sites holding the individual qubits.

We first show that deciding whether a graph state $|G\rangle$ can be transformed into another graph state $|G'\rangle$ using LC + LPM + CC is $\mathbb{NP}$-Complete, even if $|G'\rangle$ is restricted to be the GHZ-state. However, we also provide efficient algorithms for two situations of practical interest:

1. $|G\rangle$ has *Schmidt-rank width* one and $|G'\rangle$ is a GHZ-state. The Schmidt-rank width is an entanglement measure of quantum states, meaning this algorithm is efficient if the original state has little entanglement. Our algorithm has runtime $\mathcal{O}(|V(G')||V(G)|^3)$, and is also efficient in practice even on small instances as further showcased by a freely available software implementation.
2. $|G\rangle$ is in a certain class of states with unbounded Schmidt-rank width, and $|G'\rangle$ is a GHZ-state of a constant size. Here the runtime is $\mathcal{O}(\text{poly}(|V(G)|))$, showing that more efficient algorithms can in principle be found even for states holding a large amount of entanglement, as long as the output state has constant size.

Our results make use of the insight that deciding whether a graph state $|G\rangle$ can be transformed to another graph state $|G'\rangle$ is equivalent to a known decision problem in graph theory, namely the problem of deciding whether a graph G' is a *vertex-minor* of a graph G. We prove that the vertex-minor problem is $\mathbb{NP}$-Complete by relating it to a new decision problem on 4-regular graphs which we call the *semi-ordered Eulerian tour* (SOET) problem, and show that a version of the Hamiltonian cycle problem can be reduced to SOET. The SOET problem and many of the technical tools developed to obtain our results may be of independent interest.

Contents

1 Introduction

A key concept in realizing quantum technologies is the preparation of specific resource states, which then enable further quantum processing. For example, many quantum network protocols first ask to prepare a specific resource state that is shared amongst the network nodes, followed by a subsequent measurement and exchange of classical communication. The simplest instance of this concept is indeed quantum key distribution [22, 4], in which we first produce a maximally entangled state, followed by random measurements. Similarly, measurement based quantum computing [42] proceeds by first preparing the quantum device in a large resource state, followed by subsequent measurements on the qubits.

An important class of such resource states are known as graph states. These states can be described by a simple undirected and unweighted graph where the vertices correspond to the qubits of the state [30]. Apart from their broad range of applications, an appealing feature of graph states is that they can be efficiently described classically. Specifically, to describe a graph state on n qubits, only $\frac{n(n-1)}{2}$ bits are needed to specify the edges of the graph. This is in sharp contrast to the 2^n complex numbers required to describe a general quantum state [39]. It turns out that for graph states, and indeed the more general class of *stabilizer states*, their evolution under *Clifford* operations and *Pauli* measurement can be simulated efficiently on a classical computer [28].

Well known applications of graph states include cluster states [38] used in measurement based quantum computing, where together with arbitrary single-qubit measurements, these states nevertheless do form a universal resource for measurement-based quantum computation [42]. Graph states also arise as logical codewords of many error-correcting codes [45]. In the domain of quantum networking, a specific class of graph states is of particular interest. Specifically, these are states which are GHZ-like, i.e., they are equivalent to the GHZ state up to single-qubit Clifford operations. GHZ-states have been shown to be useful for applications such as quantum secret sharing [36], anonymous transfer [13], conference key agreement [43] and clock synchronization [32]. It turns out that graph states described by either a star graph or a complete graph are precisely those GHZ-states [30].

Given the desire for graph states, we may thus ask how they can effectively be prepared, and transformed. We consider the situation in which we already have a specific starting state (the *source* state), and we wish to transform it to a desired *target* state, using an available set of operations. Motivated by the fact that on a quantum network or distributed quantum processor, local operations are typically much faster and easier to implement, we consider the set of operations consisting of single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM), and classical communication (CC). Applications of an efficient algorithm that finds a series of operations to transform a source to a target state includes the ability to make effective routing - in this context, state preparation - decisions, on a distributed quantum processor or network. Here, fast decisions are essential since quantum memories are inherently noisy and the source state will therefore become useless if too much time is spent on forming a decision. Such algorithms could also be used as a design tool in the study of quantum repeater schemes [2], and the discovery of effective code switching procedures in quantum error correction [27, 37].

To understand which graph states are related under $\mathrm{LC} + \mathrm{LPM} + \mathrm{CC}$ operations we introduced the notion of a *qubit-minor* in [19]. A qubit-minor of a graph state $|G\rangle$ is another graph state $|G'\rangle$ such that $|G\rangle$ can be transformed to $|G'\rangle$ using only single-qubit Clifford operations, single-qubit Pauli measurement and classical communication. It turns out that the question of whether a graph state has a certain qubit-minor or not, can be completely phrased in graph theoretical terms. In this setting the single-qubit Clifford operations correspond to a graph operation called a *local complementation*. Consequently, the single-qubit Pauli measurements and classical communication correspond to local complementations and vertex-deletions. In graph theory, a well studied notion is that of a *vertex-minor*s of a graph, which are exactly the graphs reachable by performing local complementations and vertex-deletions. Interestingly, we show in [19] that the notion of qubit-minors are equivalent to vertex-minors, in the sense that $|G'\rangle$ is a qubit-minor of $|G\rangle$ if and only if G' is a vertex-minor of G.

Vertex-minors play an important role in algorithmic graph theory, together with the notion of *rank-width*, which is a complexity measure of graphs. Specifically, one can efficiently decide membership of a graph in some set of graphs, if this set is closed under taking vertex-minors and of fixed (bounded) rank-width [40]. An example of such a set is the set of *distance-hereditary graphs*, which are in fact exactly the graphs with rank-width one [40]. Another example of a set of graphs which is closed under taking vertex-minors are *circle graphs*, which are however of unbounded rank-width([41, Proposition 6.3] and [14]). An appealing connection between the rank-width of graphs, and the entanglement in the corresponding graph states was identified in [50], where it is shown that the rank-width corresponds to the Schmidt-rank width of the graph state, which is an entanglement measure. Specifically, the higher rank-width a graph has, the more entanglement there is in terms of this measure. Another interpretation of the *Schmidt-rank width* is that it

captures how complex the quantum state is. One way to describe quantum states is by a technique called *tree-tensor networks* and it was shown in [50] that the minimum dimension of the tensors needed to describe a state is in fact given by the *Schmidt-rank width*.

In the domain of complexity theory, the rank-width and related measures such as the tree and clique-width [16, 5] also form a measure of the inherent complexity of the underlying problem, and feature prominently in the study of fixed-parameter tractable (FPT) algorithms [21]. Specifically, a problem is called fixed-parameter tractable in terms of a parameter r, if any instance I of the problem of fixed r, is solvable in time $f(r) \cdot |I|^{\mathcal{O}(1)}$, where $|I|$ is the size of the instance and f is an arbitrary function of r [21]. In this work, the r is the rank-width, and for graphs of constant rank-width the techniques of Courcelle [17] and its generalizations [15], can be used to obtain polynomial time algorithms for problems such as Graph Coloring [24], or Hamiltonian Path [35]. While very appealing from a complexity theory point of view, however, a direct application of these techniques does however not usually lead to polynomial time algorithms that are also efficient in practice, since $f(r)$ is often prohibitively large.

Since the problem of deciding whether a graph state $|G'\rangle$ is a qubit-minor of $|G\rangle$ (QUBITMINOR) is equivalent to deciding if G' is a vertex-minor of G (VERTEXMINOR) [19], an efficient algorithm for VERTEXMINOR, directly provide an efficient algorithm for QUBITMINOR. This in turn can be used for fast decisions on how to transform graph states in a quantum network or distributed quantum processor. However, not much was previously known about the computational complexity of VERTEXMINOR and therefore if efficient algorithms exists. For a related but slightly more restrictive minor-relation, namely *pivot-minors* it has been shown in [18] that checking whether a graph G has a pivot-minor isomorphic to another graph G' is $\mathbb{NP}$-Complete. However the complexity of deciding whether G' is a vertex-minor of G was left as an open problem. What's more, we emphasize that for our application we are interested in preparing a specific target state G' on a specific set of qubits, as qubits are generally not interchangeable in the applications of our algorithm. As such, our question is not whether we can obtain a graph that is isomorphic to G', but rather whether we can prepare G' on a specific set of vertices.

Evidently, for fixed rank-width, it is not difficult to apply the techniques of Courcelle [17], to obtain an FPT algorithm for our problem that is efficient if both the size of G', as well as the rank-width of G are bounded (as we have shown in [19]). Indeed, a powerful method for deciding if a graph problem is fixed-parameter tractable is by Courcelle's theorem and its generalizations [15]. It turns out that also for our case, however, a direct implementation of Courcelle's theorem does not give an algorithm that is efficient in practice. In fact, in the case of VERTEXMINOR, this constant factor obtained by applying the techniques of Courcelle in [19] can be shown to be[1] a power of twos

$$f(r) = 2^{2^{\cdot^{\cdot^{2^r}}}} \tag{1}$$

where r is the rank-width of the input graph G and the height of the tower is 10 [19].

Results

In this paper we determine the computational complexity of VERTEXMINOR and therefore of QUBITMINOR. In particular we prove that it is in general $\mathbb{NP}$-Complete to decide whether a graph G' is a vertex-minor of another graph G. We however also give efficient algorithms for this problem whenever the input graphs belong to particular graph classes. Any overview of the complexity of the problem for different classes of graphs considered in this paper can be seen in fig. 2.

We point out that our results of $\mathbb{NP}$-Completeness and the presented algorithms also apply to the more general class of stabilizer states of relevance in quantum error correction. This is because any stabilizer state can be transformed to some graph state using only single-qubit Clifford operations. Furthermore, given a stabilizer state on n qubits, a graph state equivalent under single-qubit Clifford operations can be found efficiently in time $\mathcal{O}(n^3)$ [49].

1. We show that the problem of deciding whether a graph G' is a vertex-minor of another graph G is $\mathbb{NP}$-Complete. This implies that QUBITMINOR is also $\mathbb{NP}$-Complete. The hardness part of the above completeness result is obtained by considering a less general version of VERTEXMINOR where G' is restricted to be a star graph, which corresponds to transforming graph states to GHZ states. We call this problem STARVERTEXMINOR and show that it is also $\mathbb{NP}$-Complete, even when G is in a strict subclass of circle graphs. To show $\mathbb{NP}$-Completeness of STARVERTEXMINOR we introduce the concept of a *semi-ordered Eulerian tour* (SOET)[2]. This new graph-theoretical structure might be of independent interest. We show that the problem of deciding whether a graph

[1] Far greater than the number of atoms in the universe.
[2] Pronounced as "suit".

allows for a SOET (which we call the SOET problem) can be reduced to STARVERTEXMINOR. Furthermore, we show that deciding whether a 3-regular graph is Hamiltonian (CubHam) can be reduced to SOET. Since the CubHam has previously been shown to be $\mathbb{NP}$-Complete, this implies that VERTEXMINOR is as well, since we also show that VERTEXMINOR is in $\mathbb{NP}$. As a key technical tool we also introduce a new class of graphs which we call triangular-expanded graphs. These graphs allow us to relate Hamiltonian tours on 3-regular graphs to certain Eulerian tours on 4-regular graphs and in turn to reduce CubHam to SOET.

2. We provide an efficient algorithm for STARVERTEXMINOR, which we prove to be correct if G is distance-hereditary, or equivalently if G has rank-width one. The run-time of this algorithm is $\mathcal{O}(|V(G')||V(G)|^3)$, where $V(G)$ denotes the vertex-set of G. This algorithm can therefore be used to decide how to transform graph states, with Schmidt-rank width one, to GHZ-states using single-qubit Clifford operations, single-qubit Pauli measurements and classical communication. As mentioned above, a more general method to find efficient algorithms for certain graph problems on graphs with bounded rank-width is by using Courcelle's theorem [15]. Compared to the algorithm provided by a direct implementation of Courcelle's theorem, see [19], our algorithm presented here does not suffer from a huge constant factor in the runtime, as in eq. (1). In fact, we have implemented our efficient algorithm and see that it typically takes for example 50 ms to run for the case when $|V(G)| = 50$ on a standard desktop computer, see fig. 1. Furthermore, our algorithm is still efficient even if the size of G' is unbounded.

   Distance-hereditary graphs, and therefore graphs with rank-width one, are exactly the graphs that can be reached by adding leaves and performing twin-splits from a graph with one vertex [3]. To prove that our algorithm is correct we also present some new interesting results relating vertex-minors, distance-hereditary graphs and leaves and twins. For example we show that if v is a leaf or a twin in G but not a vertex in G', then G' is a vertex-minor of G if and only if G' is a vertex-minor of $G \setminus v$, where $\setminus v$ denotes vertex-deletion.

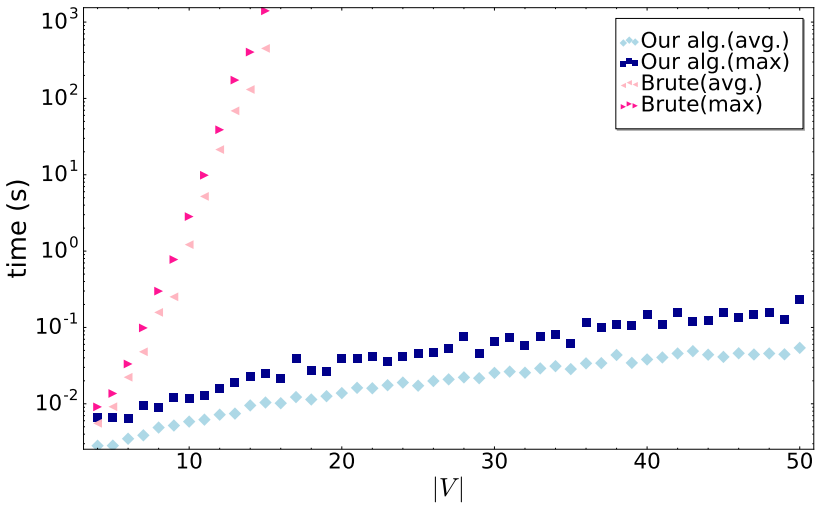


Figure 1: Average and maximal observed run-times for two algorithms that check if a star graph on four vertices is a vertex-minor of a randomly generated connected graph G on vertices V. Random connected distance-hereditary graphs are generated by starting from a single-vertex graph and randomly adding leaves or performing twin-splits, see section 2.5.1, which generates any connected distance-hereditary graph [3]. "Our alg." refers to the algorithm described in section 4.1.1 and "Brute" is the non-efficient algorithm described in [19]. The algorithm of [19] based on the techniques of Courcelle [17] is not depicted here since the pre factor makes an application impractical in practice whenever $|V| < f(r)$ of eq. (1). For each size of V, 10 random graphs are generated for "Brute" and 100 random graphs for "Our alg.", from which the average "(avg.)" and max "(max)" runtime is computed. Both algorithms are implemented in SAGE [47] and the tests were performed on an iMac with 3.2 GHz Intel Core i5 processor with 8 GB of 1600 MHz RAM.

3. We call $k-$STARVERTEXMINOR the restriction of STARVERTEXMINOR where G' is restricted to a star graph having k vertices, corresponding to a GHZ-state (up to LC) on k qubits. We show that $k-$STARVERTEXMINOR

is in $\mathbb{P}$ if G is a circle graph[3]. This means that STARVERTEXMINOR is fixed-parameter tractable in the size of G' on circle graphs. Interestingly the class of circle graphs has unbounded rank-width ([41, Proposition 6.3] and [14]) and the corresponding graph states therefore have unbounded entanglement according to the Schmidt-rank width.

4. We show that any connected graph G' on three vertices or less is a vertex-minor of any connected graph G if and only if the vertices of G' are also in G. An efficient algorithm for finding the transformation that takes the former graph to the latter is also provided.

5. We also prove several technical results that may be interesting in their own right. An example of this is theorem 4.4 which points out the interesting behavior of a certain class of graphs with respect to a bipartition of their vertices. Another example would be theorem 2.8 where we show that any distance heredity graph one more than four vertices has a *foliage* (the set of leaves, axils and twins in the graph) of size at least four.

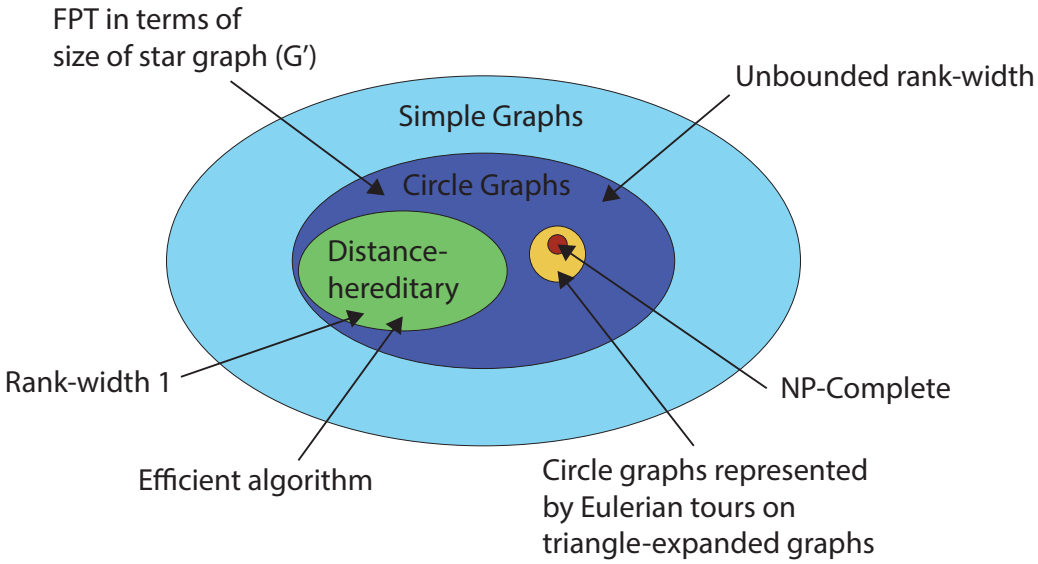


Figure 2: An overview of the graph classes discussed in this paper and what the computational complexity of solving STARVERTEXMINOR on these classes is. The sizes of the sets in the figure are not exact, however their intersections and non-intersections are. The definitions of these classes graphs are given in section 2.4 and section 2.5.1. We show that the STARVERTEXMINOR is $\mathbb{NP}$-Complete (result 1.) on graphs in a strict subclass of circle graphs (red). The same therefore holds for any superclass and in particular for circle graphs (dark blue). On the other hand we show that $k-$STARVERTEXMINOR is in $\mathbb{P}$ on circle graphs (result 3.) On distance-hereditary graphs (green) we found an efficient algorithm for STARVERTEXMINOR (result 2.) Finally it is still an open question if $k-$STARVERTEXMINOR is $\mathbb{NP}$-Complete on simple graphs (light blue) in general.

Overview

The paper is structured as follows. In section 2 we describe graph states and consider several notions of graph theory we will need throughout the paper. We also introduce the VERTEXMINOR and STARVERTEXMINOR problems and the notion of a semi-ordered Eulerian tour. We also prove a few technical results concerning distance hereditary graphs, circle graphs and vertex-minors which we will need later. Result 1. above is given in section 3, result 2. in section 4.1, result 3. in section 4.2 and result 4. in section 5.

[3]Not to be confused with cycle graph.

2 Preliminaries

In this section we set our notation and recall various concepts which will be used throughout the rest of the paper. We start by providing the definitions of graph states, qubit-minors and the relation to vertex-minors. We then recall the definitions of local complementation and vertex deletion as operations on graphs and use these to state the problems VERTEXMINOR and STARVERTEXMINOR which are the core objects of study in this paper. Furthermore, we discuss circle graphs and their various characterizations and discuss how local complementation behaves on these graphs. We also introduce the concept of semi-ordered Eulerian tours, which is a key technical concept for the results later in this paper. Finally we discuss distance-hereditary graphs, which form a subclass of circle graphs. We discuss how these graphs can be built up out of elementary pieces and prove some technical results which will be used later.

2.1 Notation and definitions

Here we introduce some notation and vocabulary that will be used throughout this paper. We assume familiarity of the general notation of quantum information theory, see [39] for more details.

Quantum operations
The Pauli matrices will be denoted as

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2}$$

The single-qubit Clifford group $\mathcal{C}$ consists of operations which leave the Pauli group $\mathcal{P} = \langle \mathrm{i}\mathbb{I}, X, Z \rangle$ invariant. More formally, $\mathcal{C}$ is the normalizer of the Pauli group, i.e.

$$\mathcal{C} = \big\{ C \in \mathcal{U} : (\forall P \in \mathcal{P} : CPC^\dagger \in \mathcal{P}) \big\}, \tag{3}$$

where $\mathcal{U}$ is the single-qubit unitary operations.

Sequences and words
A *sequence* $\boldsymbol{X} = x_1 x_2 \dots x_k$ is an ordered, possibly empty, tuple of elements in some set X. We also call a sequence a *word* and its elements *letters*. We write $\boldsymbol{X} \subseteq X$, when all letters of $\boldsymbol{X}$ are in the set X. A *sub-word* $\boldsymbol{X}'$ of $\boldsymbol{X}$, is a word which can be obtained from $\boldsymbol{X}$ by iteratively deleting the first or last element of $\boldsymbol{X}$. We denote the concatenation of two words $\boldsymbol{X}_1 = x_1 \dots x_{k_1}$ and $\boldsymbol{X}_2 = y_1 \dots y_{k_2}$ as $\boldsymbol{X}_1 \| \boldsymbol{X}_2 = x_1 \dots x_{k_1} y_1 \dots y_{k_2}$. We also denote the 'mirror image' by an overset tilde, e.g. if $\boldsymbol{X} = ab$ then $\widetilde{\boldsymbol{X}} = ba$.

Sets
The set containing the natural numbers from 1 to n is denoted $[n]$. The symmetric difference $X \Delta Y$ between two sets X and Y is the set of elements of X and Y that occur in X or Y exclusively, i.e. $X \Delta Y = (X \cup Y) \setminus (X \cap Y)$.

Graphs
A simple undirected graph $G = (V, E)$ is a set of vertices V and a set of edges E. Edges are 2-element subsets of V for simple undirected graphs. Importantly, we only consider labeled graphs, i.e. we consider a complete graph with vertices $\{1, 2, 3\}$ to be different from a complete graph with vertices $\{2, 3, 4\}$, even though these graphs are isomorphic. The reason for considering labeled graphs is that these will be used to represent graph states on specific qubits, possibly at different physical locations in the case of a quantum network. In a simple undirected graph, there are no multiple edges or self-loops, in contrast with a multi-graph: An undirected multi-graph $H = (V, E)$ is a set of vertices V and a multi-set of edges E. For undirected multi-graphs, edges are unordered pairs of elements in V. We will often write $V(G) = V$ and $E(G) = E$ to mean the vertex- and edge-set of the (multi-)graph $G = (V, E)$.

Next we list some glossary about (multi-)graphs:

- If a vertex $v \in V$ is an element of an edge $e \in E$, i.e. $v \in e$, then v and e are said to be *incident* to one another.
- Two vertices which are incident to a common edge are called *adjacent*.
- The set of all vertices adjacent to a given vertex v in a (multi-)graph G is called the *neighborhood* $N_v^{(G)}$ of v. We will sometimes just write N_v if it is clear which (multi-)graph is considered.
- The *degree* of a vertex v is the number of neighbors of v, i.e. $|N_v|$.

- A *k-regular* (multi-)graph is a (multi-)graph such that every vertex in the (multi-)graph has degree k.
- A *walk* $W = v_1 e_1 v_2 \dots e_k v_{k+1}$ is an alternating sequence of vertices and edges such that e_i is incident to v_i and v_{i+1} for $i \in [k]$.
- The vertices v_1 and v_{k+1} are called the *ends* of W.
- If the ends of a walk are the same vertex, it is called *closed*.
- A *trail* is a walk which does not include any edge twice.
- A closed trail is called a *tour*.
- A *path* is a walk which does not include any vertex twice, apart from possibly the ends.
- A closed path is called a *cycle*.
- Two vertices u and v are called *connected* if there exist a path with u and v as ends.
- A (multi-)graph is called *connected* if any two vertices are connected in the (multi-)graph.
- $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.
- An *induced subgraph* $G[V']$ of $G = (V, E)$ is a subgraph on a subset $V' \subseteq V$ and with the edge-set

$$E' = \{(u, v) \in E : u, v \in V'\}. \tag{4}$$

- A *connected component* of a (multi-)graph $G = (V, E)$ is a connected induced subgraph $G[V']$ such that no vertex in V' is adjacent to a vertex in $V \setminus V'$ in the (multi-)graph G.
- A *cut-vertex* v of a (multi-)graph $G = (V, E)$ is a vertex such that $G[V \setminus \{v\}]$ has strictly more connected components than G
- The *distance* $d_G(v, w)$ between two vertices v, w in a (multi-)graph G is equal to the number of edges in the shortest path that connects v and w.
- The *complement* G^C of a graph $G = (V, E)$ is a graph with vertex-set $V^C = V$ and edge-set

$$E^C = \{(u, v) \in V \times V : (u, v) \notin E \wedge u \neq v\}. \tag{5}$$

A graph G is assumed to be simple and undirected, unless specified and will be denoted as G, G_i, G', $\tilde{G}$ or similar. A multi-graph is assumed to be undirected, unless specified and will be denoted as H, H_i, H', $\tilde{H}$ or similar. Furthermore, 3-regular simple graphs will be denoted as R, R_i, R' or similar and 4-regular multi-graphs as F, F_i, F' or similar. We will denote the *complete graph* on a set of vertices V as K_V and the *star graph* on a set of vertices V with center c by $S_{V,c}$. We will often not care about the choice of center, writing S_V to mean any choice of star graph on the vertex set V.

2.2 Graph states

A graph state is a multi-partite quantum state $|G\rangle$ which is described by a graph G, where the vertices of G correspond to the qubits of $|G\rangle$. The graph state is formed by initializing each qubit $v \in V(G)$ in the state $|+\rangle_v = \frac{1}{\sqrt{2}}(|0\rangle_v + |1\rangle_v)$ and for each edge $(u, v) \in E(G)$ applying a controlled phase gate between qubits u and v. Importantly, all the controlled phase gates commute and are invariant under changing the control- and target-qubits of the gate. This allows the edges describing these gates to be unordered and undirected. Formally, a graph state $|G\rangle$ is given as

$$|G\rangle = \prod_{(u,v)\in E(G)} C_Z^{(u,v)} \left(\bigotimes_{v \in V(G)} |+\rangle_v \right), \tag{6}$$

where $C_Z^{(u,v)}$ is a controlled phase gate between qubit u and v, i.e.

$$C_Z^{(u,v)} = |0\rangle\langle 0|_u \otimes \mathbb{I}_v + |1\rangle\langle 1|_u \otimes Z_v \tag{7}$$

and Z_v is the Pauli-Z matrix acting on qubit v.

Any graph state is also a stabilizer state [30]. The GHZ states are an important class of stabilizer states given as

$$|\mathrm{GHZ}\rangle_k = \frac{1}{\sqrt{2}}\left(|0\rangle^{\otimes k} + |1\rangle^{\otimes k}\right). \tag{8}$$

It is easy to verify that any graph state given by a star or complete graph, i.e. $|S_{V,c}\rangle$ or $|K_V\rangle$, can be turned into a GHZ state on the qubits V using only single-qubit Clifford operations.

In the next section we discuss local complementations and vertex-deletions on graph states. It turns out that single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC): LC+LPM+CC, which take graph states to graph states, can be completely characterized by local complementations and vertex-deletions on the corresponding graphs. More concretely, any sequence of single-qubit Clifford operations, mapping graph states to graph states, can be described as some sequence of local complementations on the corresponding graph. Moreover, measuring qubit v of a graph state $|G\rangle$ in the Pauli-X, Pauli-Y or Pauli-Z basis, gives a stabilizer state that is single-qubit Clifford equivalent to $|X_v(G)\rangle$, $|Y_v(G)\rangle$, $|Z_v(G)\rangle$ respectively. The operations X_v, Y_v and Z_v are graph operations consisting of sequences of local complementations together with the deletion of vertex v, which we define in definition 2.6. As mentioned the post-measurement state of for example a Pauli-X measurement on qubit v is only single-qubit Clifford equivalent to the graph state $|X_v(G)\rangle$. The single-qubit Clifford operations that take the post-measurement state to $|X_v(G)\rangle$ depend on the outcome of the measurement of the qubit v and act on qubits adjacent to v [30]. This means classical communication is required to announce the measurement result at the vertex v to its neighboring vertices.

In [19] we introduced to notion of a *qubit-minor* which captures exactly which graph states can be reached from some initial graph state under LC + LPM + CC. Formally we define a qubit-minor as:

Definition 2.1 (Qubit-minor)**.** *Assume $|G\rangle$ and $|G'\rangle$ are graph states on the sets of qubits V and U respectively. $|G'\rangle$ is called a qubit-minor of $|G\rangle$ if there exists a sequence of single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM) and classical communication (CC) that takes $|G\rangle$ to $|G'\rangle$, i.e.*

$$|G\rangle \xrightarrow[\text{LPM+CC}]{\text{LC}} |G'\rangle \otimes |\text{junk}\rangle_{V\setminus U}. \tag{9}$$

If $|G'\rangle$ is a qubit-minor of $|G\rangle$, we denote this as

$$|G'\rangle < |G\rangle. \tag{10}$$

◇

In [19] we have shown that the notion of qubit-minors for graph states is equivalent to the notion of *vertex-minors* for graphs. We will define and discuss vertex-minors in section 2.3.1, however we formally state the relation between vertex-minors here. For a proof see [19].

Theorem 2.1 (Theorem 2.2 in [19])**.** *Let $|G\rangle$ and $|G'\rangle$ be two graph states such that no vertex in G' has degree zero. Then $|G'\rangle$ is a qubit-minor of $|G\rangle$ if and only if G' is a vertex-minor of G, i.e.*

$$|G'\rangle < |G\rangle \quad \Leftrightarrow \quad G' < G. \tag{11}$$

◇

Note that one can also include the case where G' has vertices of degree zero. Let's denote the vertices of G' which have degree zero as I. We then have that

$$|G'\rangle < |G\rangle \quad \Leftrightarrow \quad G'[V(G')\setminus I] < G. \tag{12}$$

Theorem 2.1 is very powerful since it allows us to consider graph states under LC + LPM + CC, purely in terms of vertex-minors of graphs. We will therefore in the rest of this paper use the formalism of vertex-minors to study the computational complexity of transforming graph states using LC + LPM + CC and provide efficient algorithms for transforming graph state using LC + LPM + CC.

2.3 Local complementations and vertex-deletions

Local complementation is a fundamental operation on graphs [10]. We have the following definition.

Definition 2.2 (Local complementation)**.** *A local complementation τ_v is a graph operation specified by a vertex v, taking a graph G to $\tau_v(G)$ by replacing the induced subgraph on the neighborhood of v, i.e. $G[N_v]$, by its complement. The neighborhood of any vertex u in the graph $\tau_v(G)$ is therefore given by*

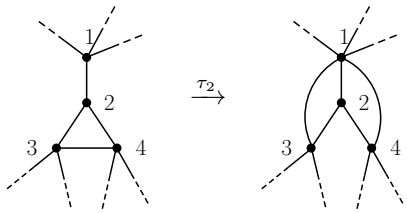
$$N_u^{(\tau_v(G))} = \begin{cases} N_u \Delta (N_v \setminus \{u\}) & \text{if } (u,v) \in E(G) \\ N_u & \text{else} \end{cases}, \tag{13}$$

where Δ denotes the symmetric difference between two sets. Given a sequence of vertices $\boldsymbol{v} = v_1 \dots v_k$, we denote the induced sequence of local complementations, acting on a graph G, as

$$\tau_{\boldsymbol{v}}(G) = \tau_{v_k} \circ \cdots \circ \tau_{v_1}(G). \tag{14}$$

◇

Below we show a simple example of the action of local complementation on a graph (in particular we consider a local complementation on the vertex labeled 2).



(15)

If two graphs G_1 and G_2 are related by a sequence of local complementations, i.e. $\exists \boldsymbol{v} : \tau_{\boldsymbol{v}}(G_1) = G_2$, we call the two graphs LC-equivalent and denote this as $G_1 \sim_{\text{LC}} G_2$. Checking whether two graphs are LC-equivalent can be done in time $\mathcal{O}(n^4)$, where n is the size of the graphs, as shown in [11]. This result was used in [48] to find an efficient algorithm for checking whether two graph states are equivalent under single-qubit Clifford operations, by proving that two graph state are equivalent under single-qubit Cliffords if and only if their corresponding graphs are LC-equivalent.

Notable about local complementation is its action on star and complete graphs. For a vertex set V and $c \in V$ we have that $\tau_c(S_{V,c}) = K_V$ and for any $v \in V$ we have $\tau_v(K_V) = S_{V,v}$. This means all star graphs on a vertex set V are equivalent to each other under local complementation and also to the complete graph on V. Moreover, no other graph is equivalent to the star or complete graph.

Another operation which we will make use of is the pivot.

Definition 2.3 (Pivot)**.** *A pivot ρ_e is a graph operation specified by an edge $e = (u,v)$, taking a graph G to $\rho_e(G)$ such that*

$$\rho_e(G) = \tau_v \circ \tau_u \circ \tau_v(G). \tag{16}$$

◇

The pivot can simply be specified by an undirected edge since

$$\tau_v \circ \tau_u \circ \tau_v(G) = \tau_u \circ \tau_v \circ \tau_u(G) \quad \text{if } (u,v) \in E(G) \tag{17}$$

as shown in [8].

It will be useful to be able to specify a pivot simply by a vertex v. We therefore also introduce the following definition:

Definition 2.4. *The graph operation ρ_v is specified by a vertex, taking a graph G to $\rho_v(G)$ such that*

$$\rho_v(G) = \begin{cases} \rho_{e_v}(G) & \text{if } |N_v| > 0 \\ G & \text{if } |N_v| = 0 \end{cases} \tag{18}$$

where e_v is an edge incident on v chosen in some consistent way. For example we could assume that the vertices of G are ordered and that $e_v = (v, \min(N_v))$. The specific choice will not matter but importantly e_v only depends on G and v, and the same therefore holds for $\rho_v(G)$. ◇

Another fundamental operation on a graph is that of vertex-deletion, which relates to measuring a qubit of a graph state in the standard basis [30]. We denote the deletion of vertex v from the graph G as $G \setminus v = G[V(G) \setminus \{v\}]$. It turns out that given a sequence of local complementations and vertex-deletions, acting on some graph, one can always perform the vertex-deletions at the end of the sequence and arrive at the same graph. This fact follows inductively from the following lemma.

Lemma 2.1. *Let $G = (V, E)$ be a graph and $v, u \in V$ be vertices such that $v \neq u$, then*

$$\tau_v(G \setminus u) = \tau_v(G) \setminus u. \tag{19}$$

◇

Proof. Note first that it is important that $v \neq u$ since the operation $\tau_v(G \setminus u)$ is otherwise undefined. To prove that the graphs $G_1 = \tau_v(G \setminus u)$ and $G_2 = \tau_v(G) \setminus u$ are equal, we show that the neighborhoods of any vertex in the graphs are the same, i.e. $N_w^{(G_1)} = N_w^{(G_2)}$ for all $w \in V(G) \setminus u$. The local complementation only changes the neighborhoods for vertices which are adjacent to v, so for any vertex $w \neq u$ which is not adjacent to v, we have that

$$N_w^{(G_1)} = N_w^{(G_2)} = N_w^{(G)} \setminus \{u\}. \tag{20}$$

On the other hand, for a vertex w which is adjacent to v, its neighborhood becomes

$$N_w^{(G_1)} = \big(N_w^{(G)} \setminus \{u\}\big)\Delta\Big(\big(N_v^{(G)} \setminus \{u\}\big) \setminus \{w\}\Big) = \Big(N_w^{(G)}\Delta\big(N_v^{(G)} \setminus \{w\}\big)\Big) \setminus \{u\} = N_w^{(G_2)} \tag{21}$$

by the definition of a local complementation. □

2.3.1 Vertex-minors

Using the two operations local complementation and vertex-deletion, we can formulate the notion of a vertex-minor of a graph.

Definition 2.5 (Vertex-minor)**.** *A graph G' is called a vertex-minor of G if and only if there exist a sequence of local complementations and vertex-deletions that takes G to G'. Since vertex-deletions can always be performed last in such a sequence (see lemma 2.1), an equivalent definition is the following: A graph G' is called a vertex-minor of G if and only if there exist a sequence of local complementations $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[V(G')] = G$. If G' is a vertex-minor of G we write this as*

$$G' < G \tag{22}$$

and if G' is not a vertex-minor of G then

$$G' \not< G. \tag{23}$$

◇

Vertex-minors were first studied in [8] but by the name of l-reductions. Note that if G_1 and G_2 are two LC-equivalent graphs, then $G' < G_1$ if and only if $G' < G_2$. It is interesting to consider under which conditions a graph G' is a vertex-minor of another graph G. As theorem 2.2 below states, to decide whether $G' < G$ it is sufficient to check whether G' is LC-equivalent to at least one out of $3^{|V(G)|-|V(G')|}$ graphs. To formally state the theorem we introduce the following three operations.

Definition 2.6. *The graph operations X_v, Y_v and Z_v, specified with a vertex v, act on a graph G by transforming it to*

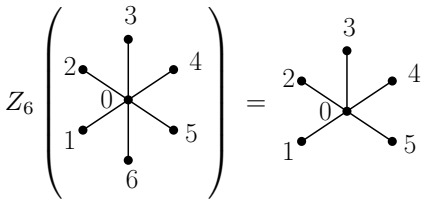
$$X_v(G) = \rho_v(G) \setminus v, \quad Y_v(G) = \tau_v(G) \setminus v, \quad Z_v(G) = G \setminus v \tag{24}$$

When we need to specify which edge incident on v the pivot of X_v acts on, we write $X_v^{(u)}(G) = \rho_{(u,v)}(G) \setminus v$. ◇

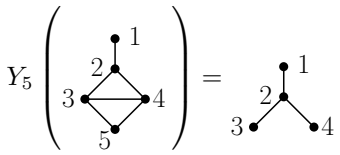
The three graph operations X_v, Y_v and Z_v correspond to how Pauli-X, -Y and -Z measurements act on graph states (as proven in [30]). As mentioned in section 2.2, measuring qubit v of a graph state $|G\rangle$ in the Pauli-X, -Y or -Z

basis gives a stabilizer state which is single-qubit Clifford equivalent to $|X_v(G)\rangle$, $|Y_v(G)\rangle$ and $|Z_v(G)\rangle$ respectively. Equations (25) to (27) shows examples of how these operations can act on graphs.

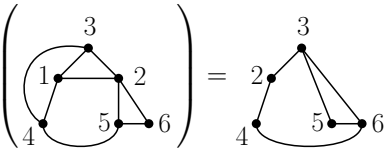
$$Z_6 \left(\begin{array}{c} \\ \\ \\ \end{array} \right. \quad = \quad \tag{25}$$



$$Y_5 \left(\begin{array}{c} \\ \\ \end{array} \right. \quad = \quad \tag{26}$$



$$X_1^{(2)} \left(\begin{array}{c} \\ \\ \\ \end{array} \right. \quad = \quad \tag{27}$$



The operation $X_v^{(u)}$ is the most complicated one, so we will here quickly describe what happens to a graph when $X_v^{(u)}$ is applied. One can check that after the operation $X_v^{(u)}$, the vertex u will have the neighbors that v previously had, except v itself. Furthermore, some edges between vertices in $(N_v \cup N_u) \setminus \{u, v\}$ will be complemented, i.e. removed if present or added if not. To know which of these edges gets complemented, let's introduce the following three sets

$$V_{vu} = N_v \cap N_u, \quad V_v = N_v \setminus (N_u \cup \{u\}), \quad V_u = N_u \setminus (N_v \cup \{v\}) \tag{28}$$

which form a partition of $(N_v \cup N_u) \setminus \{u, v\}$. In eq. (27), these sets are $V_{12} = \{3\}$, $V_1 = \{4\}$ and $V_2 = \{5, 6\}$. An edge (w_1, w_2) between vertices in $(N_v \cup N_u) \setminus \{u, v\}$ gets complemented if and only if w_1 and w_2 belong to different sets of the partition (V_{vu}, V_v, V_u). All other edges in the graph, i.e. edges containing a vertex not in $N_v \cup N_v$, will be unchanged.

It turns out that the three operations $\{X_v, Y_v, Z_v\}$ are sufficient to check whether some graph is a vertex-minor of another graph. This is formalized in the following theorem we proved in [19].

Theorem 2.2 (Theorem 3.1 in [19])**.** *Let G and G' be two graphs and let $\mathbf{u} = (v_1, \ldots, v_l)$, where $l = |V(G) \setminus V(G')|$ be an ordered tuple such that each element of $V(G) \setminus V(G')$ occurs exactly once in $\mathbf{u}$. Furthermore, let $\mathcal{P}_\mathbf{u}$ denote the set of graph operations*

$$\mathcal{P}_\mathbf{u} = \{P_{v_l} \circ \cdots \circ P_{v_1} \ : \ P_v \in \{X_v, Y_v, Z_v\}\} \tag{29}$$

Then we have that

$$G' < G \quad \Leftrightarrow \quad \exists P \in \mathcal{P}_\mathbf{u} \ : \ G' \sim_{\mathrm{LC}} P(G). \tag{30}$$

◇

Note that in [19] we indexed $\mathcal{P}_\mathbf{u}$ simply with the set associated to the word $\boldsymbol{u}$ since the statement is independent of the ordering of the elements of $\boldsymbol{u}$. A direct corollary of the above theorem is therefore:

Corollary 2.2.1. *Let G and G' be two graphs. Furthermore, let $\mathbf{u}$ and $\mathbf{u}'$ be two ordered tuples such that each element of $V(G) \setminus V(G')$ occurs exactly once in both $\mathbf{u}$ and $\mathbf{u}'$. Then we have that*

$$\exists P \in \mathcal{P}_\mathbf{u} \ : \ G' \sim_{\mathrm{LC}} P(G) \quad \Leftrightarrow \quad \exists P \in \mathcal{P}_{\mathbf{u}'} \ : \ G' \sim_{\mathrm{LC}} P(G). \tag{31}$$

◇

Proof. This follows directly from theorem 2.2 since both sides in eq. (31) are true if and only if $G' < G$. □

Note that theorem 2.2 does not give an efficient method to check if G' is a vertex-minor of G, since the set $\mathcal{P}_\mathbf{u}$ is of exponential size for all $\mathbf{u}$. To study this problem further we formally define the vertex-minor problem.

Problem 2.1 (VERTEXMINOR)**.** *Given a graph G and a graph G' defined on a subset of $V(G)$, decide whether G' is a vertex-minor of G.* ◇

Note again that we deal with labeled graphs here. We will often consider the special case where G' is a star graph $S_{V'}$ defined on a subset V' of $V(G)$. Remember that a graph state described by a star graph is single-qubit Clifford equivalent to a GHZ-state. Thus checking if $S_{V'}$ is a vertex-minor of G is equivalent to checking if $|G\rangle$ can be transformed to GHZ-state on the qubits V' by only using LC + LPM + CC. We will give this problem a separate name.

Problem 2.2 (STARVERTEXMINOR)**.** *Given a graph G and a vertex subset V' of $V(G)$, decide whether $S_{V'}$ is a vertex-minor of G.* ◇

Note that we have not specified which star graph on V' we use. This is not ambiguous since all star graphs on V' are equivalent under local complementation. In the rest of the text we will often leave the choice of star graph open.

2.4 Circle graphs

Here we introduce circle graphs and representations of these under the action of local complementations. Circle graphs are graphs with edges represented as intersections of chords on a circle. These graphs are also sometimes called alternance graphs since they can be described by a double occurrence word such that the edges of the graph are the given by the alternances induced by this word. We will make use of the latter description here, which was introduced by Bouchet in [7] and also described in [12]. This description is also related to yet another way to represent circle graphs, as Eulerian tours of 4-regular multi-graphs, introduced by Kotzig in [34]. For an overview and the history of circle graphs see for example the book by Golumbic [26].

2.4.1 Double occurrence words

Let us first define double occurrence words and equivalence classes of these. This will allow us to define circle graphs.

Definition 2.7 (Double occurrence word)**.** *A double occurrence word $\boldsymbol{X}$ is a word with letters in some set V, such that each element in V occur exactly twice in $\boldsymbol{X}$. Given a double occurrence word $\boldsymbol{X}$ we will write $V(\boldsymbol{X}) = V$ for its set of letters.* ◇

Definition 2.8 (Equivalence class of double occurrence words)**.** *We say that a double occurrence word $\boldsymbol{Y}$ is equivalent to another $\boldsymbol{X}$, i.e. $\boldsymbol{Y} \sim \boldsymbol{X}$, if $\boldsymbol{Y}$ is equal to $\boldsymbol{X}$, the mirror $\widetilde{\boldsymbol{X}}$ or any cyclic permutation of $\boldsymbol{X}$ or $\widetilde{\boldsymbol{X}}$. We denote by $\boldsymbol{d_X} = \{\boldsymbol{Y} : \boldsymbol{Y} \sim \boldsymbol{X}\}$ the equivalence class of $\boldsymbol{X}$, i.e. the set of words equivalent to $\boldsymbol{X}$.* ◇

Next we define alternances of these equivalence classes, which will represent the edges of an alternance graph.

Definition 2.9 (Alternance)**.** *An* alternance *(u, v) of the equivalence class $\boldsymbol{d_X}$ is a pair of distinct elements $u, v \in V$ such that a double occurrence word of the form $\ldots u \ldots v \ldots u \ldots v \ldots$ is in $\boldsymbol{d_X}$.* ◇

Note that if (u, v) is an alternance of $\boldsymbol{d_X}$ then so is (v, u), since the mirror of any word in $\boldsymbol{d_X}$ is also in $\boldsymbol{d_X}$.

Definition 2.10 (Alternance graph)**.** *The alternance graph $\mathcal{A}(\boldsymbol{X})$ of a double occurrence word $\boldsymbol{X}$ is a graph with vertices $V(\boldsymbol{X})$ and edges given exactly by the alternances of $\boldsymbol{d_X}$, i.e.*

$$E(\mathcal{A}(\boldsymbol{X})) = \{(u, v) \in V(\boldsymbol{X}) \times V(\boldsymbol{X}) : (u, v) \text{ is an alternance of } \boldsymbol{d_X}\} \tag{32}$$

◇

Note that since $\mathcal{A}(\boldsymbol{X})$ only depends on the equivalence class of $\boldsymbol{X}$, the alternance graphs $\mathcal{A}(\boldsymbol{X})$ and $\mathcal{A}(\boldsymbol{Y})$ are equal if $\boldsymbol{X} \sim \boldsymbol{Y}$. Now we can formally define[4] circle graphs.

Definition 2.11 (Circle graph)**.** *A graph G which is the alternance graph of some double occurrence word $\boldsymbol{X}$ is called a circle graph.* ◇

[4] Circle graphs are usually defined as graphs which edges are represented by intersections of chords on a circle. However, the definition in terms of alternances of double occurrence words turn out to be equivalent [12].

2.4.2 Eulerian tours on 4-regular multi-graphs

There is yet another way to represent circle graphs, closely related to double occurrence words, as Eulerian tours of 4-regular multi-graphs.

Definition 2.12 (Eulerian tour)**.** *Let F be a connected 4-regular multi-graph. An Eulerian tour U on F is a tour that visits each edge in F exactly once.* ◇

Any 4-regular multi-graph is Eulerian, i.e. has a Eulerian tour, since each vertex has even degree [6].

Furthermore, any Eulerian tour on a 4-regular multi-graph F traverses each vertex exactly twice, except for the vertex which is both the start and the end of the tour. Such a Eulerian tour induces therefore a double occurrence word, the letters of which are the vertices of F, and consequently a circle graph as described in the following definition.

Definition 2.13 (Induced double occurrence word)**.** *Let F be a connected 4-regular multi-graph on k vertices $V(F)$. Let U be a Eulerian tour on F of the form*

$$U = x_1 e_1 x_2 \dots x_{2k-1} e_{2k-1} x_{2k} e_{2k} x_1. \tag{33}$$

with $x_i \in V$. Note that every element of V occurs exactly twice in U. From a Eulerian tour U as in eq. (33) we define an induced double occurrence word as

$$m(U) = x_1 x_2 \dots x_{2k-1} x_{2k}. \tag{34}$$

To denote the alternance graph given by the double occurrence word induced by a Eulerian tour, we will write $\mathcal{A}(U) \equiv \mathcal{A}(m(U))$. ◇

Similarly to double occurrence words, we also introduce equivalence classes of Eulerian tours under cyclic permutation or reversal of the tour.

Definition 2.14 (Equivalence class of Eulerian tours)**.** *Let F be a connected 4-regular multi-graph and U be an Eulerian tour on F. We say that an Eulerian tour U' on F is equivalent to U, i.e. $U \sim U'$, if U' is equal to U, the reversal $\widetilde{U}$ or any cyclic permutation of U or $\widetilde{U}$. We denote by $\boldsymbol{t}_U$ the equivalence class of U, i.e. the set of Eulerian tours on F which are equivalent to U.* ◇

It is clear that if the Eulerian tours U and U' on a 4-regular multi-graph F are equivalent, then so are the double occurrence words $m(U)$ and $m(U')$. Furthermore, as for double occurrence words, two equivalent Eulerian tours on a connected 4-regular multi-graph induce the same alternance graph.

2.4.3 Local complementations on circle graphs

We will now introduce an operation $\widetilde{\tau}_v$ on double occurrence words that will be the equivalent of performing a local complementation on the corresponding alternance graph.

Definition 2.15 ($\widetilde{\tau}_v$)**.** *Let $\boldsymbol{X}$ be a double occurrence word and v be an element in $V(\boldsymbol{X})$. We can then always find sub-words $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ not containing v, such that $\boldsymbol{X} = \boldsymbol{A}v\boldsymbol{B}v\boldsymbol{C}$. Note that some of the sub-words $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are possibly empty. The operation $\widetilde{\tau}_v$ acting on a double occurrence word is then defined as*

$$\widetilde{\tau}(\boldsymbol{A}v\boldsymbol{B}v\boldsymbol{C}) = \boldsymbol{A}v\widetilde{\boldsymbol{B}}v\boldsymbol{C}. \tag{35}$$

If $\mathbf{v} = (v_1, \dots, v_l)$ is a sequence of elements of $V(\boldsymbol{X})$ we use the notation $\widetilde{\tau}_{\mathbf{v}}(\boldsymbol{X}) = \widetilde{\tau}_{v_l} \circ \cdots \circ \widetilde{\tau}_{v_1}(\boldsymbol{X})$. ◇

The operation $\widetilde{\tau}_v$ in the above definition maps equivalence classes to equivalence classes, as defined in definition 2.8. That is, if $\boldsymbol{X} \sim \boldsymbol{Y}$ and $v \in V(\boldsymbol{X})$, then $\widetilde{\tau}_v(\boldsymbol{X}) \sim \widetilde{\tau}_v(\boldsymbol{Y})$. For example, assume that $\boldsymbol{Y}$ is the mirror of $\boldsymbol{X}$, i.e. $\boldsymbol{Y} = \widetilde{\boldsymbol{X}}$. Then we see that

$$\widetilde{\tau}_v(\boldsymbol{X}) = \boldsymbol{A}v\widetilde{\boldsymbol{B}}v\boldsymbol{C} \sim \widetilde{\boldsymbol{A}v\widetilde{\boldsymbol{B}}v\boldsymbol{C}} = \widetilde{\boldsymbol{C}}v\boldsymbol{B}v\widetilde{\boldsymbol{A}} = \widetilde{\tau}_v(\boldsymbol{Y}). \tag{36}$$

The case when $\boldsymbol{Y}$ is a obtained by a cyclic permutation of $\boldsymbol{X}$ can be checked similarly.

In [12] it was shown that the alternance graph of $\mathcal{A}(\widetilde{\tau}_v(\boldsymbol{X})$, where $\boldsymbol{X}$ is a double occurrence word and $v \in V(\boldsymbol{X})$, is the same as the graph obtained by performing a local complementation at v, i.e.

$$\tau_v(\mathcal{A}(\boldsymbol{X})) = \mathcal{A}(\widetilde{\tau}_v(\boldsymbol{X})). \tag{37}$$

Similar to the above we can also define an operation $\bar{\tau}_v$ on Eulerian tours U on 4-regular multi-graphs which also has the effect of a local complementation on the graph $\mathcal{A}(U)$.

Definition 2.16 ($\bar{\tau}_v$)**.** *Let F be a connected 4-regular multi-graph. Let U be a Eulerian tour on F and v be a vertex in V. Let P_v be the first subtrail of U that starts and ends at v, i.e. $U = U_1 P_v U_2$, from some U_1 and U_2. We define $\bar{\tau}_v(U)$ to be the Eulerian tour obtained by traversing U_1, the reversal of P_v and then U_2, i.e. $\bar{\tau}_v(U) = U_1 \widetilde{P_v} U_2$. When $\mathbf{v} = v_1 \dots v_l$ is a sequence of vertices in V we write $\bar{\tau}_{\mathbf{v}}(U) \equiv \bar{\tau}_{v_l} \circ \cdots \circ \bar{\tau}_{v_1}(U)$.* ◇

Note in particular that $\bar{\tau}_v(U)$, where U is an Eulerian tour on F, is also a Eulerian tour on F.

We have now defined τ-operations on circle graphs, $\widetilde{\tau}$-operations on double occurrence words and $\bar{\tau}$-operations on Eulerian tours of 4-regular multi-graphs. They are given similar names since they are in some sense the same operation but in different representations of circle graphs. To see this note that

$$m(\bar{\tau}_v(U)) = m(U_1 \widetilde{P_v} U_2) = \widetilde{\tau}_v(m(U)) \tag{38}$$

where $U = U_1 P_v U_2$ as in definition 2.16. From eq. (37) and the shorthand $\mathcal{A}(U) = \mathcal{A}(m(U))$ we also have that

$$\mathcal{A}(\bar{\tau}_v(U)) = \mathcal{A}(\widetilde{\tau}_v(m(U))) = \tau_v(\mathcal{A}(U)). \tag{39}$$

The operation $\bar{\tau}_v$ on Eulerian tours of 4-regular multi-graphs was introduced by Kotzig in [33], where he called it a κ-transformation.

As stated by Bouchet in [12], Kotzig [33] proved that any two Eulerian tours of a 4-regular multi-graph are related by a sequence of κ-transformations.

Theorem 2.3 (Proposition 4.1 in [12], [33])**.** *Let U and U' be Eulerian tours on the same connected 4-regular multi-graph. Then there exist a sequence $\mathbf{v}$ such that $\tau_{\mathbf{v}}(U) \sim U'$.* ◇

2.4.4 Vertex-deletion on circle graphs

When we are considering vertex-minors of circle graphs, it is useful to have an operation on the double occurrence word that corresponds to the deletion of a vertex in the corresponding alternance graph. Let $\boldsymbol{X} = \boldsymbol{A}v\boldsymbol{B}v\boldsymbol{C}$ be a double occurrence word and v be an element in $V(\boldsymbol{X})$. We will denote by $\boldsymbol{X} \setminus v$ the deletion of the element v, i.e.

$$\boldsymbol{X} \setminus v \equiv (\boldsymbol{A}v\boldsymbol{B}v\boldsymbol{C}) \setminus v = \boldsymbol{ABC}. \tag{40}$$

The resulting word $\boldsymbol{ABC}$ is also a double occurrence word and furthermore we have that

$$\mathcal{A}(\boldsymbol{X}) \setminus v = \mathcal{A}(\boldsymbol{X} \setminus v). \tag{41}$$

If $W = \{w_1, w_2 \dots, w_l\}$ is a subset of V, we will write $\boldsymbol{X} \setminus W$ as the deletion of all elements in W, i.e.

$$\boldsymbol{X} \setminus W = (\dots((\boldsymbol{X} \setminus w_1) \setminus w_2) \dots) \setminus w_l. \tag{42}$$

Connected to this we can also define an induced double occurrence sub-word $m[W] = \boldsymbol{X} \setminus (V \setminus W)$. The reason for calling this an induced double occurrence sub-word stems from its relation to induced subgraphs of the alternance graph as

$$\mathcal{A}(\boldsymbol{X})[W] = \mathcal{A}(\boldsymbol{X}[W]). \tag{43}$$

2.4.5 Vertex-minors of circle graphs

Since we now have expressions for local complementation and vertex deletion on circle graphs in terms of double occurrence words, we can consider vertex-minors of circle graphs completely in terms of double occurrence words. More precisely we have the following theorem.

Theorem 2.4. *Let G be a circle graph such that $G = \mathcal{A}(\boldsymbol{X})$ for some double occurrence word $\boldsymbol{X}$. Then G' is a vertex-minor of G if and only if there exist a sequence $\mathbf{v}$ of elements in $V(G) = V(\boldsymbol{X})$ such that*

$$G' = \mathcal{A}(\widetilde{\tau}_{\mathbf{v}}(\boldsymbol{X})[V(G')]). \tag{44}$$

◇

Proof. By using eq. (43) and eq. (39) on the right hand side of eq. (44) we have that

$$\mathcal{A}\Big(\widetilde{\tau}_{\mathbf{v}}(\boldsymbol{X})[V(G')]\Big) = \mathcal{A}\Big(\widetilde{\tau}_{\mathbf{v}}(\boldsymbol{X})\Big)[V(G')] = \tau_{\mathbf{v}}(\mathcal{A}(\boldsymbol{X}))[V(G')] \tag{45}$$

Since G' is a vertex-minor of $G = \mathcal{A}(\boldsymbol{X})$ if and only if there exist a sequence $\mathbf{v}$ of elements in $V(G)$ such that

$$G' = \tau_{\mathbf{v}}(G)[V(G')] \tag{46}$$

the theorem follows. □

We can also consider vertex minors of circle graphs in terms of their representations as Eulerian tours on connected 4-regular multi-graphs, which we will use in section 3 to prove that VERTEXMINOR is $\mathbb{NP}$-Complete. Theorem 2.3, together with eq. (39), implies that connected 4-regular multi-graphs describe equivalence classes of circle graphs under local complementations. Bouchet pointed out this fact in [12]. We formalize this here as a theorem together with a formal proof:

Theorem 2.5. *Let U be an Eulerian tour of a connected 4-regular multi-graph F with vertices V. Then (1) any graph LC-equivalent to $\mathcal{A}(U)$ is an alternance graph of some Eulerian tour of F and (2) any alternance graph of a Eulerian tour of F is a graph LC-equivalent to $\mathcal{A}(U)$.* ◇

Proof. We start by proving (1), so let us therefore assume that G is a graph LC-equivalent to $\mathcal{A}(U)$. This means, by definition, that there exist a sequence $\mathbf{v}$ of vertices in V such that $G = \tau_{\mathbf{v}}(\mathcal{A}(U))$. By using eq. (39) we have that

$$G = \mathcal{A}(\bar{\tau}_{\mathbf{v}}(U)). \tag{47}$$

which shows that G is an alternance graph induced by a Eulerian tour of F, since $\bar{\tau}_{\mathbf{v}}(U)$ is a Eulerian tour on F. To prove (2), assume that U' is a Eulerian tour of F. We will now prove that the alternance graph of U', $\mathcal{A}(U')$, is LC-equivalent to $\mathcal{A}(U)$. By theorem 2.3, we know that there exist a sequence of $\bar{\tau}_v$-transformations that relates U and U', i.e. there exist a sequence $\mathbf{v}$ such that

$$\bar{\tau}_{\mathbf{v}}(U) \sim U'. \tag{48}$$

Since these Eulerian tours are equivalent, their induced alternance graphs are equal, i.e.

$$\mathcal{A}(\bar{\tau}_{\mathbf{v}}(U)) = \mathcal{A}(U'). \tag{49}$$

Finally, using eq. (39) on the above equation gives

$$\tau_{\mathbf{v}}(\mathcal{A}(U)) = \mathcal{A}(U') \tag{50}$$

which shows that $\mathcal{A}(U)$ and $\mathcal{A}(U')$ are indeed LC-equivalent. □

Similarly to theorem 2.4 we can decide if a circle graph has a certain vertex-minor by considering Eulerian tours of a 4-regular graph, which is captured in the following theorem.

Theorem 2.6. *Let F be a connected 4-regular multi-graph and let G be a circle graph such that $\mathcal{A}(U)$ for some Eulerian tour U on F. Then G' is a vertex-minor of G if and only if there exist a Eulerian tour U' on F such that*

$$G' = \mathcal{A}(m(U')[V(G')]). \tag{51}$$

◇

Proof. Lets first assume that G' is a vertex-minor of G. This means that there exists a sequence $\mathbf{v}$ such that $G' = \tau_{\mathbf{v}}(G)[V(G')]$. Since $G = \mathcal{A}(U)$ we have that

$$G' = \tau_{\mathbf{v}}\Big(\mathcal{A}(U)\Big)[V(G')] \tag{52}$$

$$= \mathcal{A}\Big(\bar{\tau}_v(U)\Big)[V(G')] \tag{53}$$

$$= \mathcal{A}\Big(m(\bar{\tau}_v(U))[V(G')]\Big) \tag{54}$$

where we used eq. (39) in the first step and eq. (43) in the second. We therefore see that $U' = \bar{\tau}_v(U)$ is a Eulerian tour on F satisfying eq. (51).

To prove the converse let us assume that there exist a Eulerian tour U' on F satisfying eq. (51). From theorem 2.3 we know that there exist a sequence $\mathbf{v}$ such that $U' = \bar{\tau}_{\mathbf{v}}(U)$. We can then replace U' by $\bar{\tau}_{\mathbf{v}}(U)$ in eq. (51) such that

$$G' = \mathcal{A}\Big(m(\bar{\tau}_{\mathbf{v}}(U))[V(G')]\Big) \tag{55}$$

$$= \mathcal{A}\Big(m(\bar{\tau}_{\mathbf{v}}(U))\Big)[V(G')] \tag{56}$$

$$= \tau_{\mathbf{v}}\Big(\mathcal{A}(U)\Big)[V(G')] \tag{57}$$

where we again made use of eq. (43) and eq. (39). From eq. (57) we see that G' is indeed a vertex-minor of G, see definition 2.5, since $G = \mathcal{A}(U)$. □

2.4.6 Semi-Ordered Eulerian tours

From the previous sections we have seen that circle graphs and their vertex-minors can be described by Eulerian tours on connected 4-regular multi-graphs. As discussed in section 2.2, the question of whether a given graph G has vertex-minors on the subset $V' \subseteq V(G)$ in the form of star or complete graphs is of importance in quantum information theory, since it corresponds to transforming the graph state $|G\rangle$ to a GHZ-state on the qubits V'. A natural question is therefore: Given a set of vertices V', what property should a connected 4-regular multi-graph F satisfy, such that $S_{V'}$ is a vertex-minor of $\mathcal{A}(U)$, for some Eulerian tour U on F. As we will see in this section, a necessary and sufficient condition is that F allows for what we call a *semi-ordered Eulerian tour* (SOET) with respect to V'.

The existence of a SOET on a four regular graph F with respect to some vertex set V' will therefore be a key technical tool when considering STARVERTEXMINOR on circle graphs, as described in section 3. We formally define a SOET as follows.

Definition 2.17 (SOET)**.** *Let F be a 4-regular multi-graph and let $V' \subseteq V(F)$ be a subset of its vertices. Furthermore, let $\boldsymbol{s} = s_1 s_2 \dots s_k$ be a word with letters in V' such that each element of V' occurs exactly once in $\boldsymbol{s}$ and where $k = |V'|$. A semi-ordered Eulerian tour U with respect to V' is a Eulerian tour such that $m(U) = \boldsymbol{X}_0 s_1 \boldsymbol{X}_1 s_2 \dots s_k \boldsymbol{X}_k s_1 \boldsymbol{Y}_1 s_2 \dots s_k \boldsymbol{Y}_k$ for some $\boldsymbol{s}$ and where $\boldsymbol{X}_0, \boldsymbol{X}_1, \dots, \boldsymbol{X}_k, \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_k$ are words (possibly empty) with letters in $V \setminus V'$. This can also be stated as $m(U)[V'] = \boldsymbol{s}\boldsymbol{s}$, for some $\boldsymbol{s}$.* ◇

Note that the multi-graph F is not assumed to be simple, so multi-edges and self-loops are allowed. A SOET is a Eulerian tour on F that traverses the elements of V' in some order once and then again in the same order. The particular order in which the SOET traverses V' will not be important here, only that it traverses V' in the same order twice. An example of a graph that allows for a SOET with respect to the set $V' = \{a, b, c, d\}$ can be seen in fig. 3a. A SOET for this graph is for example $m(U) = abcdaebced$. The graph in fig. 3b on the other hand does not allow for any SOET with respect to the set $V' = \{a, b, c, d\}$.

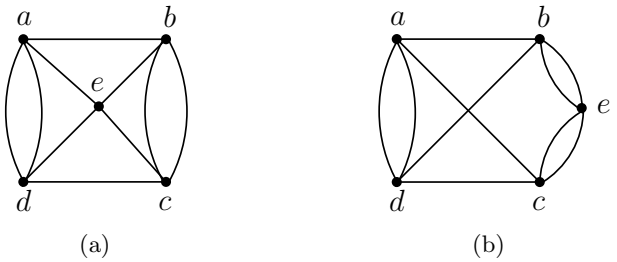


Figure 3: Examples of two 4-regular multi-graphs. The graph in fig. 3a allows for a SOET with respect to the set $\{a, b, c, d\}$ but the graph in fig. 3b does not.

We also formally define the SOET-decision problem, which takes a 4-regular multi-graph F and a subset V' of the vertices as input and asks to decide whether or not the graph F allows for a Semi-Ordered Eulerian Tour with respect to the vertex set V'.

Problem 2.3 (SOET)**.** *Let F be a 4-regular multi-graph and let V' be a subset $V(F)$. Decide whether there exists a* SOET *U on F with respect to the set V'.* ◇

As mentioned, the reason for introducing the notion of a SOET is that a 4-regular multi-graph F allows for a SOET with respect to a subset $V' \subseteq V(F)$ if and only if a star graph on V' is a vertex-minor of an alternance graph $\mathcal{A}(U)$ induced by a Eulerian tour U on F. This is captured in the following theorem, formulated as a corollary of theorem 2.6.

Corollary 2.6.1. *Let F be a connected 4-regular multi-graph and let G be a circle graph given by the alternance graph of a Eulerian tour U on F, i.e. $G = \mathcal{A}(U)$. Then $S_{V'}$ is a vertex-minor of G if and only if F allows for a* SOET *(see definition 2.17) with respect to V'.* ◇

Proof. Note first that $S_{V'} \leq G$ if and only if $K_{V'} \leq G$, since $S_{V'}$ and $K_{V'}$ are LC equivalent. From theorem 2.6 we know that $K_{V'}$ is a vertex-minor of G if and only if there exist an Eulerian tour U' on F such that

$$K_{V'} = \mathcal{A}(m(U')[V']). \tag{58}$$

It is easy to verify that $\mathcal{A}(\boldsymbol{X})$ is a complete graph on V' if and only if $\boldsymbol{X} = s_1 s_2 \dots s_k s_1 s_2 \dots s_k$ where $\boldsymbol{s} = s_1 s_2 \dots s_k$ is a word with letters in V' such that each element of V' occur exactly once in $\boldsymbol{s}$. The result then follows, since $m(U')[V']$ is of this form if and only if U' is a SOET with respect to V'. □

One can see that the existence of a SOET on a 4-regular multi-graph F with respect to V', imparts an ordering on the subset of vertices V'. We will in particular be interested in vertices in V' that are 'consecutive' with respect to the SOET. Consecutiveness is defined as follows.

Definition 2.18 (Consecutive vertices)**.** *Let F be a 4-regular graph and U a* SOET *on F with respect to a subset $V' \subseteq V(F)$. Two vertices $u, v \in V'$ are called consecutive in U if there exist a sub-word $u\boldsymbol{X}v$ or $v\boldsymbol{X}u$ of $m(U)$ such that no letter of $\boldsymbol{X}$ is in V'.* ◇

We also define the notion of a "maximal sub-word" associated with two consecutive vertices.

Definition 2.19 (Maximal sub-words)**.** *Let F be a 4-regular multi-graph and U a SOET on F with respect to a subset $V' \subseteq V(F)$. The double occurrence word induced by U is then of the form $m(U) = \boldsymbol{X}_0 s_1 \boldsymbol{X}_1 s_2 \dots s_k \boldsymbol{X}_k s_1 \boldsymbol{Y}_1 s_2 \dots s_k \boldsymbol{Y}_k$, where $k = |V'|$, $s_1, \dots, s_k \in V'$ and $X_0, \dots, X_k, Y_1, \dots, Y_k$ are words (possibly empty) with letters in $V(F) \setminus V'$. For $i \in [k-1]$, we call $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ the two maximal sub-words associated with the consecutive vertices s_i and s_{i+1}. Furthermore, we call $\boldsymbol{X}_k$ and $\boldsymbol{Y}_k \boldsymbol{X}_0$ the two maximal sub-words associated with the consecutive vertices s_k and s_1. Given two consecutive vertices u and v, we will denote their two maximal sub-words as $\boldsymbol{X}$ and $\boldsymbol{X}'$, $\boldsymbol{Y}$ and $\boldsymbol{Y}'$ or similar.* ◇

2.5 Leaves, twins and axils

In this section we will consider certain vertices called leaves, twins and axils. First we will prove that such vertices can in many cases be removed when considering the vertex-minor problem, which can simplify the problem significantly. We capture this in theorem 2.7. This motivates us to consider distance-hereditary graphs, since it turns out that these are exactly the graphs that can be reached from a single-vertex graph by adding leaves or performing twin-splittings. We will leverage these properties in section 4.1 to find an efficient algorithm for STARVERTEXMINOR when the input graph is distance hereditary. We define and consider distance-hereditary graphs in section 2.5.1.

Let us first formally define leaves, twins and axils.

Definition 2.20 (Leaves and axils)**.** *A* leaf *is vertex with degree one. An* axil *is the unique neighbor of a leaf.* ◇

Definition 2.21 (Twin)**.** *A* twin *is a vertex v such that there exist a different vertex u with the same neighborhood, i.e. v is a twin if and only if*

$$\exists u \in V \setminus \{v\} : \big(N_v \setminus \{u\} = N_u \setminus \{v\}\big). \tag{59}$$

A vertex u as in eq. (59) is called a twin-partner *of v and v, u form a* twin-pair*. If v and u are adjacent, they form a* true twin-pair *and otherwise a* false twin-pair*.* ◇

Definition 2.22 (Foliage)**.** *The* foliage *of a graph G is the set of leaves, axils and twins in a graph G and is denoted*

$$T(G) = \{v \in V(G) : v \text{ is a leaf, axil or twin}\} \tag{60}$$

$\diamond$

We are now ready to prove the following theorem which can be used to simplify some instances of VERTEXMINOR, in particular when considering distance-hereditary graphs, see section 2.5.1.

Theorem 2.7. *Let G and G' be graphs and v be a vertex in G but not in G'. Then the following is true:*

- *If v is a leaf or a twin, then G' is a vertex-minor of G if and only if G' is a vertex-minor of $G \setminus v$, i.e.*

$$G' < G \quad \Leftrightarrow \quad G' < (G \setminus v). \tag{61}$$

- *If v is an axil, then G' is a vertex-minor of G if and only if G' is a vertex-minor of $\tau_w \circ \tau_v(G) \setminus v$, where w is the leaf associated to v, i.e.*

$$G' < G \quad \Leftrightarrow \quad G' < (\tau_w \circ \tau_v(G) \setminus v). \tag{62}$$

$\diamond$

Proof. Firstly, if G' is a vertex-minor of $G \setminus v$, then clearly G' is also a vertex-minor of G.

This means we only need to prove the other direction. Assume therefore that G' is a vertex-minor of G. We start by proving the case where v is a leaf in G. The cases where v is an axil or a twin in G then follow by a short argument.

Hence assume that v is a leaf in $V \setminus V'$, where $V = V(G)$ and $V' = V(G')$. Furthermore, let $\boldsymbol{u}$ be a sequence of vertices such that each element of $V \setminus V'$ occurs exactly once in $\boldsymbol{u}$. Since G' is a vertex-minor of G, we know by theorem 2.2 that there exists some sequence of operations $P \in \mathcal{P}_{\boldsymbol{u}}$, such that $P(G) \sim_{\mathrm{LC}} G'$. Let us denote the i-th operation in P as $P^{(i)}$, such that $P = P^{(n-k)} \circ \cdots \circ P^{(1)}$, where $n = |G|$ and $k = |G'|$. Remember that each operation $P^{(i)}$ deletes the i-th vertex of $\boldsymbol{u}$ from the graph. Furthermore, let's denote the sequence of operations from i through j in P as

$$P_i^j = P^{(j)} \circ \dots P^{(i+1)} \circ P^{(i)}. \tag{63}$$

By corollary 2.2.1 we know that such a P exist for all orderings $\boldsymbol{u}$ of the vertices in $V \setminus V'$. Without loss of generality we can assume that v is the first element in $\boldsymbol{u}$. This means that $P^{(1)}$ is either Z_v, Y_v or X_v. We will now treat all three these cases separately.

If $P^{(1)}$ is Z_v or Y_v, then since v is a leaf we have that

$$P^{(1)}(G) = G \setminus v. \tag{64}$$

Then it is easy to see that G' is also a vertex-minor of $G \setminus v$, since

$$G' \sim_{\mathrm{LC}} P(G) = P_2^{n-k} \circ P^{(1)}(G) = P_2^{n-k}(G \setminus v) \tag{65}$$

If $P^{(1)}$ is X_v then the axil of v cannot be in $V \setminus V'$, since the operation X_v on a leaf disconnects the axil from its neighbors. Lets denote the axil of v by w and assume again w.l.o.g. that the ordering of $V \setminus V'$ is such that w is the second element of $\boldsymbol{u}$. Since w is a disconnected vertex after $P^{(1)}$, any of the three operations $\{X_w, Y_w, Z_w\}$ act the same, i.e. deleting w. So the action of X_v followed by $P^{(2)} \in \{X_w, Y_w, Z_w\}$ is the same as deleting both v and w or in other words

$$P_1^2(G) = Z_w(G \setminus v) \tag{66}$$

It is again clear that G' is then a vertex-minor of $G \setminus v$, since

$$G \sim_{\mathrm{LC}} P(G) = P_3^{n-k} \circ P_1^2(G) = P_3^{n-k} \circ Z_w(G \setminus v) \tag{67}$$

with a satisfying sequence taking $G \setminus v$ to an LC-equivalent graph of G' being $(Z_w, P^{(3)}, \dots, P^{(n-k)})$. This proves the theorem when v is a leaf.
Now assume that v is a twin in G. To prove that the theorem also hold for twins, we first show that a twin can

always be transformed into a leaf by local complementations. Assume that v and w are false twins, and denote one of their common neighbors as n.[5] Then the graph $\tilde{G} = \tau_w \circ \tau_n(G)$ is a graph where v is a leaf and w is an axil. Since LC-equivalent graphs have the same vertex-minors, G' is also a vertex-minor of $\tilde{G}$. From what we showed above and that v is a leaf, G' is also a vertex-minor of $\tilde{G} \setminus v$. Finally, G' is then also a vertex-minor of

$$\tau_n \circ \tau_w(\tilde{G} \setminus v) = \tau_n \circ \tau_w \circ \tau_w \circ \tau_n(G) \setminus v = G \setminus v \tag{68}$$

where we used lemma 2.1. An almost identical argument can be made for the case where v and w are true twins by considering the graph $\tilde{G} = \tau_w(G)$.

Now assume v is an axil in G. If v is an axil in G and $v \notin G'$, then v is a leaf in the graph $\tilde{G} = \tau_w \circ \tau_v(G)$, where w is the leaf of v in G. Since by assumption $G' < G$, we know that $G' < \tilde{G}$ and from the cases of leaves we have that also $G' < \tilde{G} \setminus v$, since v is a leaf in $\tilde{G}$. This completes the proof. □

2.5.1 Distance-hereditary graphs

In this section we introduce distance-hereditary graphs. As shown by Bouchet in [9], distance-hereditary graphs are exactly the graphs with rank-width one. These graphs have nice properties which we make use of in section 4.1.

Definition 2.23 (Distance-hereditary)**.** *A graph G is distance-hereditary if and only if, for each connected induced subgraph $G[A]$ and for any two vertices $u, v \in A$ the distance between u and v is the same in G and in $G[A]$, i.e.*

$$d_G(u, v) = d_{G[A]}(u, v). \tag{69}$$

◇

The simplest example of a graph which is not distance-hereditary is the five-cycle C_5. To see this pick two vertices which have distance two in C_5 and denote their unique common neighbor by v. The distance between the same vertices in the connected induced subgraph $C_5[V \setminus v]$ is three and thus not the same as in C_5. It turns out that distance-hereditary graphs are exactly the graphs which do not contain a vertex-minor isomorphic to C_5 [12]. We also note that distance-hereditary graphs form a strict subclass of circle graphs [12].

In [3] an equivalent property of distance-hereditary is shown: A graph is distance-hereditary if and only if it can be obtained from a single-vertex graph using the following three operations:

- *Add a leaf*: Let u be a vertex in a graph G. Add the vertex v and the edge (u, v) to G.
- *False twin-split*: Let u be a vertex in a graph G. Add the vertex v and the edges $\{(v, x) : x \in N_u\}$.
- *True twin-split*: Let u be a vertex in a graph G. Add the vertex v and the edges $\{(v, x) : x \in \{u\} \cup N_u\}$.

Note that this implies that a distance-hereditary graph always has at least one leaf or twin, i.e. the foliage is non-empty. This fact will be a critical element of the algorithm presented in section 4.1

In the rest of this section we prove some properties of the foliage for distance-hereditary graphs, which we make use of in section 4.1 to find an efficient algorithm for VERTEXMINOR on distance-hereditary graphs. First we show that the twin relation is in fact transitive. This is a technical lemma we will use in later theorems.

Lemma 2.2. *Let G be a graph and let u be a vertex of G that is a twin and has twin-partners $\{t_1, t_2, \ldots, t_k\}$. Then all vertices in $u \cup \{t_1, t_2, \ldots, t_k\}$ are pairwise twins.* ◇

Proof. Since u and t_i form a twin-pair, for $i \in \{1, 2, \ldots, k\}$, we have that

$$N_u \setminus \{t_i\} = N_{t_i} \setminus \{u\} \tag{70}$$

which implies that

$$N_{t_i} = (N_u \setminus \{t_i\}) \cup \{u\}. \tag{71}$$

[5] Note that twins always have at least one common neighbor, except for the graph K_2 where the twins are anyway also leaves.

Thus, we have that

$$N_{t_i} \setminus \{t_j\} = ((N_u \setminus \{t_i\}) \cup \{u\}) \setminus \{t_j\} \tag{72}$$
$$= (\underbrace{(N_u \setminus \{t_j\})}_{N_{t_j} \setminus \{u\}} \cup \{u\}) \setminus \{t_i\} \tag{73}$$
$$= N_{t_j} \setminus \{t_i\}. \tag{74}$$

This shows that, for $i \neq j$, t_i and t_j form a twin-pair. □

Next we prove that adding leaves to a graph G or performing (true or false) twin-splits never decreases the size of the foliage $T(G)$.

Lemma 2.3. *Assume G is a connected, distance-hereditary graph. Let G' be a graph formed by doing a twin-split on G or adding a leaf to G. Then*

$$|T(G')| \geq |T(G)|, \tag{75}$$

where $T(G)$ is the foliage of G. ◇

Proof. To prove this, let us first consider the case when $|G| \leq 2$. Since G is connected it is necessary the case that $G = K_1$ or $G = K_2$:

- If $G = K_1$, then $G' = K_2$ and $|T(G')| = 2 \geq 0 = |T(G)|$.
- If $G = K_2$, then $G' = K_3$ or $G' = P_3$ and $|T(G')| = 3 \geq 2 = |T(G)|$.

Let's now consider the case when $|G| > 2$. We consider the two cases when G' is formed by adding a leaf and performing a twin-split separately:

- Assume G' is formed by adding a leaf v to G, making u an axil of G'.Note first that if $u \notin T(G)$, then $|T(G)|$ can only increase since no vertex in $T(G)$ was affected. Let's therefore assume that $u \in T(G)$. There are then three possibilities: (1) u is a leaf, (2) u is an axil but not a twin and (3) u is a twin. We consider these three cases separately:
  - (1) Assume u is a leaf in G. Then the axil of u in G, is not in $T(G')$, but both u and v are. Therefore $|T(G)| = |T(G')|$.
  - (2) Assume u is an axil but not a twin in G. Then u is also an axil in G' and we have that $|T(G')| = |T(G)| = 1$.
  - (3) Assume u is a twin in G.
    * Assume there is only one twin-pair containing u in G. Then the twin-partner of u in G, is not in $T(G')$, but both u and v are. Therefore $|T(G')| = |T(G)|$.
    * Assume there is more than one twin-pair containing u in G. Then the twin-partners of u are all pairwise twins, by lemma 2.2, and will still be in G'. Therefore $|T(G')| = |T(G)| + 1$.
- Assume G' is formed by twin-splitting u in G, creating v and making v and u a twin-pair. Note first that if $u \notin T(G)$, then $|T(G)|$ can only increase since no vertex in $T(G)$ was affected. Let's therefore assume that $u \in T(G)$. There are then three possibilities: (1) u is a leaf, (2) u is an axil but not a twin and (3) u is a twin. We consider these three cases separately:
  - (1) Assume u is a leaf in G. Then the axil of u in G is either still an axil in G' or not, depending on if u and v are true or false twins. In either case, $|T(G')| \geq |T(G)|$ since $v \in T(G')$.
  - (2) Assume u is an axil but not a twin in G. Note that all leaves with u as an axil in G are also twins. These vertices are also twins in G' since they are all now also adjacent to v. Thus, $|T(G')| = |T(G)| + 1$.
  - (3) Assume u is a twin in G.
    * Assume there is only one twin-pair in G containing u. Then the twin-partner of u in G, may or may not still be a twin-partner of u in G' depending on whether the considered twin-pairs are true or false. Therefore the size of $T(G)$ either remains the same or increases by one since again $v \in T(G')$.

* Assume there are more than one twin-pair in G containing u. Then the twin-partners of u are all pairwise twins, by lemma 2.2, and will still be in G'. Therefore $|T(G')| = |T(G)| + 1$.

□

We now make use of the above theorem to prove that the foliage has a certain minimum size.

Theorem 2.8. *Assume G is a connected, distance-hereditary graph and $2 \leq k \leq 4$, then*

$$|G| \geq k \qquad \Rightarrow \qquad |T(G)| \geq k. \tag{76}$$

◇

Proof. First we explicitly check that the graphs on 2, 3 and 4 vertices has $T(G) = 2$, $T(G) = 3$ and $T(G) = 4$, respectively.[6] Then by lemma 2.3 and the fact that all distance-hereditary graphs can be built up by twin-splits and adding leaves [9], the result follows. □

We point out that the theorem does not hold for $k > 4$. Consider for example a path graph P_k on more than four vertices. It is easy to see that size of the foliage in this case is $|P_k| = 4$.

Finally we show that an interesting property regarding the foliage, in relation to cut-vertices.[7]

Corollary 2.8.1. *Assume that G is a connected distance-hereditary graph and that $v \in G$ is a cut-vertex. Denote the connected components of $G \setminus v$ by $G_1, G_2, \ldots, G_k$, where k is the number of connected components of $G \setminus v$. Then for any $1 \leq i \leq k$, there exist a vertex $u \in G_i$ such that $u \in T(G)$.* ◇

Proof. Pick an arbitrary connected component G_i with vertices V_i. If G_i is just a single vertex, then this vertex is necessarily a leaf in G and is therefore in $T(G)$. Now assume that $|G_i| > 1$, then by using theorem 2.8, we have that there exist at least one twin-pair not containing v or a leaf which is not v in $G[\{v\} \cup V_i]$. This proves the corollary. □

3 Complexity

In this section we will consider the time-complexity of VERTEXMINOR. In particular we will show that even a highly restrictive version of the vertex-minor problem is ℕℙ-Hard, namely when G' is a star graph and G is in a strict subclass of circle graphs. Since we also prove that VERTEXMINOR is in ℕℙ this then proves that VERTEXMINOR is ℕℙ-Complete.

3.1 VERTEXMINOR is in ℕℙ

We begin by arguing that the vertex-minor problem is in ℕℙ. Given graphs G and G' such that $G' < G$, a witness to this relation would be a sequence of local complementations and vertex deletions that takes G to G'. It is not a priori clear that this sequence is polynomial in length w.r.t. to the number of vertices of G. However from theorem 2.2 one can argue that whenever there is such a sequence, there is also a sequence of polynomial length. This leads to the following theorem.

Theorem 3.1. *The decision problem* VERTEXMINOR *is in* ℕℙ. ◇

Proof. Let G and G' be graphs, on n and k vertices respectively. Furthermore, let $\boldsymbol{u}$ be a sequence such that each element of $V(G) \setminus V(G')$ occur exactly once in $\boldsymbol{u}$. If $G' < G$ then, by theorem 2.2, there exists a sequence of operations $P \in \mathcal{P}_{\boldsymbol{u}}$, as specified in eq. (29), such that $P(G) \sim_{\mathrm{LC}} G'$. Furthermore, the sequence of operations P consists of $\mathcal{O}(n-k)$ local complementations and vertex-deletions. A witness to the instance (G, G') will then be the sequence of operations P. On the other hand, if $G' \not< G$, then by theorem 2.2, there exist no $P \in \mathcal{P}_{\boldsymbol{u}}$ such that $P(G) \sim_{\mathrm{LC}} G'$.

[6] The number of non-isomorphic graphs on 2, 3 and 4 vertices are 1, 2 and 6, respectively.
[7] A cut-vertex is a vertex such that when it is deleted, the number of connected components increases.

Given (G, G') and a sequence of operations P, a verifier can therefore perform the following protocol to check if (G, G') is a *yes*-instance of STARVERTEXMINOR.

1. Compute $P(G)$.
2. Decide if $P(G) \sim_{\text{LC}} G'$ using Bouchet's algorithm for checking if two graphs are LC-equivalent [11].
3. Output *yes* if Bouchet's algorithm outputs TRUE and *no* otherwise.

The verifier will therefore output *yes* if P is such that $P(G) \sim_{\text{LC}} G'$ and *no* if $G' \not\sim G$, since then $P(G) \nsim_{\text{LC}} G'$ for any P. Computing $P(G)$ can be done in time $\mathcal{O}(n^2(n-k))$, since each local complementation can be performed in time $\mathcal{O}(n^2)$ [11]. Furthermore, checking whether $P(G)$ and G' are LC-equivalent can be done in time $\mathcal{O}(k^4)$ using Bouchet's algorithm [11]. Thus the verifier will output *yes* or *no* in time $\mathcal{O}(n^2(n-k)) + \mathcal{O}(k^4)$. □

3.2 VERTEXMINOR is NP-Complete

Next we will argue that the problem VERTEXMINOR is also NP-Hard and hence that it is NP-Complete. We will do this through a sequence of three reductions.

- First we will reduce STARVERTEXMINOR to VERTEXMINOR. This is done in theorem 3.2.
- Secondly we will reduce a new problem, which we call the SOET problem, for Semi-Ordered Eulerian Tour, to STARVERTEXMINOR. This is done in section 3.2.1
- Finally we will reduce the problem of deciding whether a 3-regular (or cubic) graph has a *Hamiltonian cycle* or not (CubHam), which is a known NP-complete problem [25], to the SOET problem. This is the most complicated part of the reduction and is done in several steps in section 3.2.2.

The reductions between these problems are summarized in fig. 4. Eventually we will have the following theorem, which can be considered the main theorem of this section.

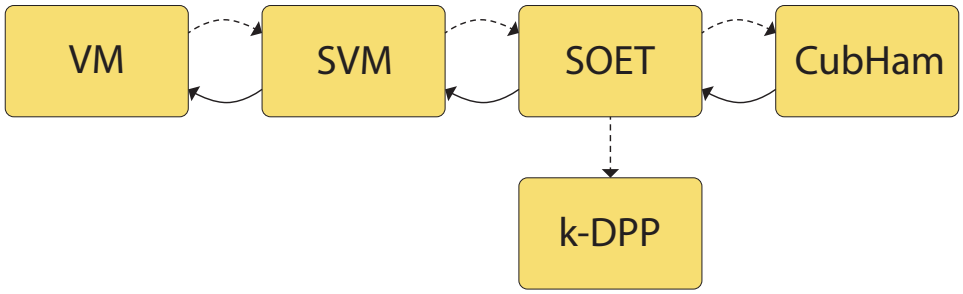


Figure 4: The different decision problems considered and how they can be reduced to each other. An solid arrow between problem A and B means that any instance of A can be reduced to B in polynomial time. A dashed line means that a subclass of instances of the A can be reduced to B in polynomial time. VERTEXMINOR (VM) and STARVERTEXMINOR (SVM) are presented in section 2.3.1, SOET in section 2.4.6, CubHam in section 3.2 and $k-$DPP (k-Disjoint Paths Problem) in section 4.2. The dashed line from STARVERTEXMINOR to SOET is restricted to circle graphs. The dashed line from SOET to CubHam is restricted to triangle-expanded graphs $\Lambda(R)$ where the SOET is with respect to $V(R)$. Finally the dashed line from SOET to $k-$DPP is for SOET problems where the SOET is with respect to a set of size k.

Theorem 3.2. VERTEXMINOR *is* NP*-Complete* ◇

Proof. Note first that STARVERTEXMINOR trivially reduces to VERTEXMINOR. This is so since every *yes*(*no*)-instance of STARVERTEXMINOR is also an *yes*(*no*)instance of VERTEXMINOR. From theorem 3.3 we see that we can reduce the SOET problem to STARVERTEXMINOR and finally from corollary 3.4.1 we see that we can reduce CubHam to the SOET problem. Since CubHam is a known NP-Complete problem this implies that VERTEXMINOR is NP-Hard. From theorem 3.1 we have that VERTEXMINOR is in NP and hence it is NP-Complete. □

Now we will detail every step in the above reduction. We begin with proving that the SOET decision problem reduces to STARVERTEXMINOR.

3.2.1 Reducing the SOET problem to STARVERTEXMINOR

In this section we show that the SOET problem reduces to STARVERTEXMINOR. For this we will make use of the properties of circle graphs, discussed in section 2. In corollary 2.6.1 we showed that a 4-regular multi-graph F allows for a SOET with respect to a subset of its vertices $V' \subseteq V(F)$ if and only if an alternance graph $\mathcal{A}(U)$ (which is a circle graph), induced by some Eulerian tour on F, has $S_{V'}$ as a vertex-minor.

Since circle graphs are a subset of all simple graphs we can then decide whether a 4-regular graph F allows for a SOET with respect to some subset V' of it's vertices by constructing the circle graph induced by an Eulerian tour on F and checking whether it has a star-vertex minor on the vertex set V'. This leads to the following theorem.

Theorem 3.3. *The decision problem* SOET *reduces to* STARVERTEXMINOR. ◇

Proof. Let (F, V') be an instance of SOET, where F is a 4-regular multi-graph and V' a subset of the vertex set of F. Also let G be a circle graph induced by any Eulerian tour U on F. From corollary 2.6.1 we see that G has $S_{V'}$ as a vertex-minor if and only if F allows for a SOET with respect to the vertex set V'. Since an Eulerian tour U can be found in polynomial time [23] and since G can be efficiently constructed given U, this concludes the reduction. □

3.2.2 Reducing CubHam to the SOET problem

In this section we will prove that the SOET problem, as defined in problem 2.3, is $\mathbb{NP}$-Complete by reducing the problem of deciding if a 3-regular graph is Hamiltonian (CubHam), a well-known $\mathbb{NP}$-Complete problem [25], to the SOET problem (it is in $\mathbb{NP}$ by theorem 3.3 and theorem 3.1). For completeness we include the definition of a Hamiltonian graph.

Definition 3.1 (Hamiltonian)**.** *A graph is said to be Hamiltonian if it contains a Hamiltonian cycle. A Hamiltonian cycle is a cycle that visits each vertex in the graph exactly once.* ◇

We can use this to formally define the CubHam problem.

Problem 3.1 (CubHam)**.** *Let R be a 3-regular graph. Decide whether R is Hamiltonian.* ◇

The reduction of CubHam to the SOET problem is done by going though the following steps.

1. Introduce the notion of a (4-regular) triangular-expansion $\Lambda(R)$ of a 3-regular graph. This is done in definition 3.2.
2. Argue that given a 3-regular graph R, its triangular-expansion can be constructed efficiently. This is done in lemma 3.1.
3. Introduce the notions of *skip* and *true skip* that capture an essential behavior of SOETs on triangular-expansions of 3-regular graphs. This is done in section 3.2.4.
4. Prove that if a 3-regular graph R is Hamiltonian then the triangular-expansion $\Lambda(R)$ of R allows for a SOET with respect to the set $V(R)$. This is done in lemma 3.3.
5. Prove that if the triangular-expansion $\Lambda(R)$ of a 3-regular graph R allows for a special kind of SOET, called a HAMSOET with respect to R (HAMSOETs are defined in definition 3.4, but can be thought of as SOETs with no true skips), then the 3-regular graph R is Hamiltonian. This is done in lemma 3.4.
6. Prove that if the triangular-expansion $\Lambda(R)$ of a 3-regular graph R allows for a SOET with respect to $V(R)$ then it also allows for a HAMSOET with respect to R. This is done in lemma 3.5.

Performing all these steps will lead to the following theorems.

Theorem 3.4. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion as defined in definition 3.2. R is Hamiltonian if and only if $\Lambda(R)$ allows for a* SOET *with respect to $V(R)$.* ◇

Proof. Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion as defined in definition 3.2. If R is Hamiltonian then lemma 3.3 guarantees that $\Lambda(R)$ allows for a SOET with respect to the vertices $V(R)$. In the other direction, if $\Lambda(R)$ allows for a SOET with respect to the vertices $V(R)$ then we can see from lemma 3.4 that it also allows for a HAMSOET. The existence of a HAMSOET on $\Lambda(R)$ then implies, via lemma 3.4 that R has a Hamiltonian cycle and hence that it is Hamiltonian. This proves the theorem. □

Corollary 3.4.1. *The SOET problem is* $\mathbb{NP}$*-Complete.* ◇

Proof. The Hamiltonian cycle problem (CubHam) is $\mathbb{NP}$-Complete on 3-regular graphs [25]. We will reduce this problem to the SOET problem. Let R be an instance of CubHam, i.e. a 3-regular graph. From this 3-regular graph we can construct its triangular-expansion $\Lambda(R)$. In lemma 3.1 it is argued that this construction can be performed in $O(|V(R)|)$ time. We can then use theorem 3.4 to see that R is Hamiltonian if and only if $\Lambda(R)$ allows for a SOET with respect to the vertex set $V(R)$. Hence there exists an efficient reduction of CubHam to the SOET problem. This means that the SOET problem is $\mathbb{NP}$-Hard. Furthermore, the SOET problem is in $\mathbb{NP}$ since it can be efficiently reduced to STARVERTEXMINOR by theorem 3.3, which is in $\mathbb{NP}$ by theorem 3.1. Hence the SOET problem is $\mathbb{NP}$-Complete. □

Corollary 3.4.2. *The* SOET *problem is* $\mathbb{NP}$*-Complete on graphs which are triangular-expansions of planar 3-regular triply-connected graphs, i.e. graphs in the set*

$$\{\Lambda(R) : R \text{ is planar, 3-regular and triply-connected}\}. \tag{77}$$

◇

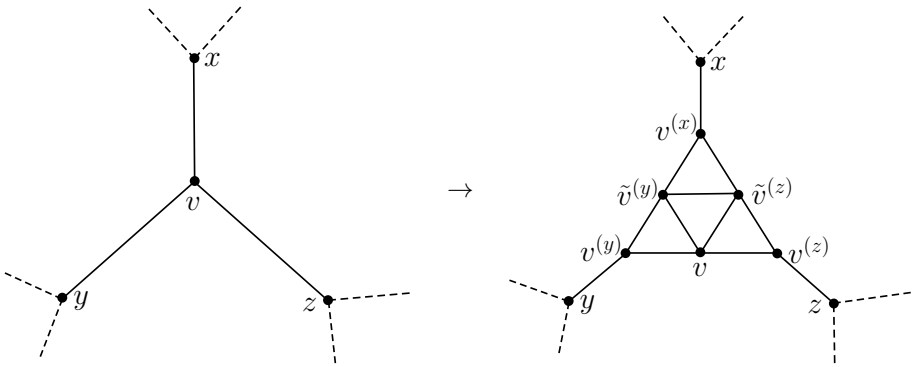
Proof. The proof is the same as the proof of corollary 3.4.1 but using the fact that CubHam is $\mathbb{NP}$-Complete on planar triply-connected graphs. □

3.2.3 triangular-expansions

It now remains to prove lemmas 3.3 to 3.5. These lemmas will relate Hamiltonian cycles on 3-regular graphs and SOETs on 4-regular multi-graphs by using a mapping from 3-regular graphs to 4-regular multi-graphs. We call this mapping 'triangular-expansion'. We have the following definition.

Definition 3.2 (Triangular-expansion)**.** *Let R be a* 3*-regular graph. A triangular-expansion $\Lambda(R)$ of a* 3*-regular graph R is constructed from R by performing the following two steps:*

1. *Replace each vertex v in R with the subgraph below*



(78)

   where x, y and z are the neighbors of v. We will denote this triangle subgraph *associated to the vertex v with T_v, i.e. $T_v = G[\{v, v^{(x)}, v^{(y)}, v^{(z)}, \tilde{v}^{(y)}, \tilde{v}^{(z)}\}]$.*

2. *Double every edge that is incident on two subgraphs $T_v, T_{v'}$.*

◇

The graph $\Lambda(R)$ will be called a *triangular-expansion* of R. A multi-graph F that is the triangular-expansion of some 3-regular graph R will also be referred to as a triangular-expanded graph. Note that the triangular-expansion is not uniquely defined, since for each vertex $v \in R$ there is a choice how to orient the triangle with respect to the neighbors of v. Furthermore, the number of vertices in $\Lambda(R)$ is $6 \cdot |V(R)|$ and the number of edges is $2 \cdot |E(R)| + 9 \cdot |V(R)|$. In fig. 5 we show an example of a 3-regular graph and its triangular-expansion.

For a given triangle subgraph T_v in a triangular-expanded graph, we will refer to the vertices adjacent to other triangle subgraphs T_x, T_y, T_z as 'outer vertices' and label them according to the triangle subgraph they are adjacent to. Concretely we label the vertex in T_v that is adjacent to T_w as $v^{(w)}$, the index signifies which triangle subgraph it connects to.

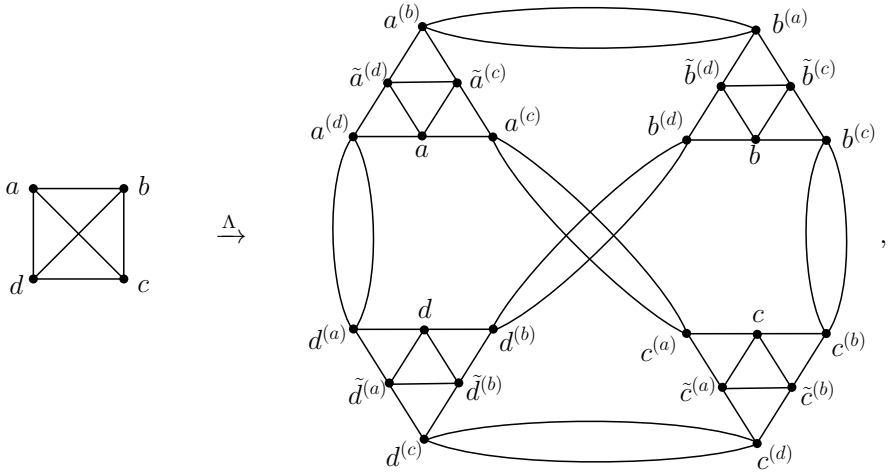


Figure 5: Figure showing the complete graph on vertices $V = \{a, b, c, d\}$ and its associated triangular-expansion $\Lambda(K_V)$.

In the following lemma we argue that this construction can be made efficiently in the size of R.

Lemma 3.1. *Let R be a 3-regular graph. We can construct its triangular-expansion in $O(|V(R)|^2)$ time.* ◇

Proof. Let R be a 3-regular graph. Without loss of generality assume some labeling on the vertices of R, i.e. $V(R) = \{v_1, \ldots, v_k\}$ where $k = |V(R)|$. We begin by constructing the vertex set $V(\Lambda(R))$ off the triangular-expansion $\Lambda(R)$ of R.

For each $i \in [k]$ construct the set $V_i = \{v_i, \tilde{v}_i^{(v_j)}, \tilde{v}_i^{(v_{j'})}, v_i^{(v_j)}, v_i{}^{(v_{j'})}, v^{(v_{\hat{j}})}\}$ where $v_i \in V(R)$ and $v_j, v_{j'}, v_{\hat{j}}$ are the three unique neighbors of v_i in R. Constructing this set takes $O(|V(R)|)$ as we must search the set of edges $E(R)$ of R to find the neighbors of v_i and we have that $|E(R)| = O(|V(R)|)$ since R is 3-regular. Thus constructing the set $V(\Lambda(R)) = \cup_{i \in [k]} V_i$ takes $O(|V(R)|^2)$ time.

Now we construct the edge multi-set $E(\Lambda(R))$ of the triangular-expansion of R. For each $i \in [k]$ we define the multi-set

$$E_i = \{(v_i, \tilde{v}_i^{(v_j)}), (v_i, \tilde{v}_i^{(v_{j'})}), (v_i, v_i^{(v_j)}), (v_i, v_i^{(v_{j'})}), (\tilde{v}_i^{(v_j)}, \tilde{v}_i^{(v_{j'})}), (v_i^{(v_j)}, \tilde{v}_i^{(v_j)}), (v_i^{(v_{j'})}, \tilde{v}_i^{(v_{j'})}), \tag{79}$$

$$(v_i^{(v_{\hat{j}})}, \tilde{v}_i^{(v_j)}), (v_i^{(v_{\hat{j}})}, \tilde{v}_i^{(v_{j'})}), (v_i^{(v_j)}, v_j^{(v_i)}), (v_i^{(v_{j'})}, v_{j'}^{(v_i)}), (v_i^{(v_{\hat{j}})}, v_{\hat{j}}^{(v_i)})\}. \tag{80}$$

This multi-set can be constructed in constant time. Hence the multi-set $E(\Lambda(R)) = \cup_{i \in [k]} E_i$ can be constructed in $O(|V(R)|)$ time. It is easy to check that the multi-graph defined by the vertex set $\Lambda(R)$ and edge multi-set $E(\Lambda(R))$ is indeed the triangular-expansion of R. This completes the lemma.

□

3.2.4 Skips and true skips

A key insight in the behavior of the SOET problem on triangular-expanded graphs is the notion of *skips*. The word skip stems from the fact that since any SOET U on the triangular-expansion $\Lambda(R)$ of a 3-regular graph is a Eulerian tour, it must traverse any triangle subgraph T_v of $\Lambda(R)$ exactly three times. However in order for U to be a valid SOET with respect to $V(R)$ it must traverse the vertex v exactly two of those three times. This means it must *skip* the vertex v exactly once while traversing T_v.

We state a more formal definition of a (true) skip in terms of maximal sub-words (see definition 2.19).

Definition 3.3 (Skip)**.** *Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. Let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. Let $\boldsymbol{X}$ be a maximal sub-word of $m(U)$ (definition 2.19) associated to vertices $u, v \in V(R)$. We say the sub-trail described by $\boldsymbol{X}$ makes a* skip *at a vertex $w \in V(R) \setminus \{u, v\}$ if $x_1^{(w)} w^{(x_1)}$ and $x_2^{(w)} w^{(x_2)}$ are sub-words of $\boldsymbol{X}$ (up to reflection), where $x_1, x_2 \in V(R)$. Furthermore, if $x_1 \neq x_2$ then we say that the trail described by $\boldsymbol{X}$ makes a true skip at w or sometimes that T_w contains a* true skip. ◇

Note that since $\boldsymbol{X}$ is a maximal sub-word associated to u, v and $w \notin \{u, v\}$, w cannot be a letter of $\boldsymbol{X}$. As stated above, there is always exactly one maximal sub-word describing a sub-trail of a SOET that makes a skip at a certain triangle subgraph, as formalized in the following lemma. One can think of this lemma as giving necessary conditions for the existence of a SOET with respect to $V(R)$ on the triangular-expansion of a 3-regular graph R

Lemma 3.2. *Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. Let U be a Eulerian tour on $\Lambda(R)$. Let $w \in V(R)$ and let T_w be its triangle subgraph in $\Lambda(R)$. If U is a* SOET *on $\Lambda(R)$ with respect to $V(R)$ then here exist exactly one maximal sub-trail of U that makes a skip at w.* ◇

Proof. We will prove this by showing that there are exactly three maximal sub-trails of U that traverse vertices in T_w and that exactly one of these makes a skip at w. Note first that the Eulerian tour U will enter and exit the triangle subgraph T_w exactly three times, since there are six edges incident to T_w. Hence there exists exactly three distinct edge-disjoint sub-trails, t_1, t_2 and t_3 of U that exit and enter T_w, i.e. the last vertices they traverse in T_w will be $x_i^{(w)}$, where x_i for $i \in [3]$ are the neighbors of w in R. Note that t_1, t_2 and t_3 each contain at least one vertex in T_w and they jointly traverse all edges in T_w (Since U is a Eulerian tour).

Now consider the vertex w. The Eulerian tour U traverses this vertex exactly twice. There are now two options for the trails t_1, t_2, t_3. Either (1) one of the trails contains the vertex w exactly twice or (2) there are exactly two trails that contain the vertex w exactly once.

Now assume U is a SOET with respect to $V(R)$. If option (1) is true the tour U traverses the vertex w twice in succession before traversing any other vertex in the set V'. This is in contradiction with the assumption that U is a SOET. Hence if U is a SOET we must have that option (2) is true, that is, exactly two of the sub-trails t_1, t_2 and t_3 must contain w.

Lets assume without loss of generality that $w \in t_2$ and $w \in t_3$. Hence we have that $w \notin t_1$ and thus t_1 induces a sub-word $m(t_1)$ not containing w. The word $m(t_1)$ can be extended to a maximal sub-word $\boldsymbol{X}$ not associated to w but describing a sub-trail traversing vertices in T_w. Therefore by definition 3.3 $\boldsymbol{X}$ describes a sub-trail making a skip at w. Furthermore, $m(t_2)$ has an overlap with two maximal sub-words associated with y_1, w and w, y_2, respectively for some $y_1, y_2 \in V(R)$. Similarly for $m(t_3)$ and the vertices z_1, w and w, z_2. Thus, the maximal sub-words that have an overlap with $m(t_2)$ or $m(t_3)$ do not make a skip at w.

Finally, there is no other maximal sub-word describing a sub-trail that traverses a vertex in T_w. The reason for this is that t_1, t_2 and t_3 jointly traverse all edges in T_w, so any maximal sub-trail traversing a vertex in T_w must have an overlap with t_1, t_2 or t_3. The lemma then follows since we found exactly one maximal sub-word describing a sub-trail of U making a skip at w, i.e. the unique one that has an overlap with t_1. □

3.2.5 Equivalence between SOET's and Hamiltonian cycles

Now that we have defined the triangular-expansion $\Lambda(R)$ of a 3-regular graph R and discussed skips we can finally make the central argument of the reduction given in corollary 3.4.1. We begin by proving that if a 3-regular graph is Hamiltonian then its corresponding triangular-expansion $\Lambda(R)$ allows for a SOET w.r.t. the vertex set $V(R)$. We have the following lemma.

Lemma 3.3. *Let R be a 3-regular graph and $\Lambda(R)$ be a triangular-expansion as defined in definition 3.2. If R is Hamiltonian, then $\Lambda(R)$ allows for a SOET with respect to $V(R)$.* ◇

Proof. Let R be Hamiltonian. This means there exists a Hamiltonian cycle M on R. We will prove that there exists a SOET U with respect to $V(R)$ on the triangular-expansion $\Lambda(R)$ of R by constructing, from the cycle M on R, a tour U that visits every vertex $v \in V(R)$ twice in the same order. We will then argue that this tour can always be lifted to a Eulerian tour and hence can be made into a SOET.

Note first that M induces an ordering on the vertices of R which without of loss of generality we will take to be $v_1, \dots v_k$ where $k = |V(R)|$. Note that for all $i \in [k-1]$ the vertices v_i and v_{i+1} are adjacent in R and so are v_k and v_1. Now consider, for each $i \in [k-1]$ the triangle subgraphs T_{v_i}, $T_{v_{i+1}}$ and $T_{\hat{v}}$ in the triangular-expansion $\Lambda(R)$ of R where $\hat{v}_i$ is the unique vertex adjacent to v_i in R that is not v_{i+1} or v_{i-1}. There are now three cases, depending on the orientation of the triangle subgraph T_{v_i}. Either (1) v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(v_{i+1})}$ or (2) v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(\hat{v}_i)}$ or (3) v_i is adjacent to $v_i^{(v_{i+1})}$ and $v_i^{(\hat{v}_i)}$.

For case (1), i.e. v_i is adjacent to $v_i^{(v_{i-1})}$ and $v_i^{(v_{i+1})}$ in T_{v_i}, we can define two edge-disjoint trails on the triangular-expansion $\Lambda(R)$ by their description as words $\boldsymbol{X}_i, \boldsymbol{X}'_i$ on the vertices of $\Lambda(R)$.

$$\boldsymbol{X}_i = v_i^{(v_{i-1})} \tilde{v}_i^{(v_{i-1})} v_i^{(\hat{v}_i)} \tilde{v}_i^{(v_{i+1})} v_i v_i^{(v_{i+1})} v_{i+1}^{(v_i)}, \tag{81}$$

$$\boldsymbol{X}'_i = v_i^{(v_{i-1})} v_i \tilde{v}_i^{(v_{i-1})} \tilde{v}_i^{(v_{i+1})} v_i^{(v_{i+1})} v_{i+1}^{(v_i)}. \tag{82}$$

An illustration of the trails described by these words is given in fig. 6a. We can similarly define two edge-disjoint trails for the cases (2) and (3). We will abuse notation and refer to the words as $\boldsymbol{X}_i$ and $\boldsymbol{X}'_i$ in all three cases. Importantly, for all three these cases the trails described by $\boldsymbol{X}_i, \boldsymbol{X}'_i$ are edge-disjoint and both begin at $v_i^{(v_{i-1})}$ and end at $v_i^{(v_{i+1})}$. We can extend this definition to trails U_0, U'_0 also for the case of the adjacent vertices v_k and v_1. Note now that the walk U described by the word $\boldsymbol{W} = \boldsymbol{X}_0 \boldsymbol{X}_{\boldsymbol{1}} \dots \boldsymbol{X}_{k-1} \boldsymbol{X}'_0 \boldsymbol{X}'_1 \dots \boldsymbol{X}_{k-1}$ is a tour and moreover that it traverses the vertices in $V(R)$ twice in the order $v_1, \dots v_k$.

The tour U described by the word $\boldsymbol{W}$ above visits every vertex in $V(R)$ such that $\boldsymbol{W}[V(R)] = v_1 \dots v_k v_1 \dots v_k$. However it is not yet a Eulerian tour, since it has not traversed all edges in $\Lambda(R)$. Note that the edges not traversed by U are precisely the triangular-expansions of the edges in R not traversed by the Hamiltonian cycle M. We can easily construct a Eulerian tour out of U by looping over all elements of the word $\boldsymbol{X}$ and whenever we encounter a vertex $v_i^{(\hat{v}_i)}$ for all $i \in [k]$, where $\hat{v}_i$ is defined as above, we check whether U already traverses the edges $(v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)})$. If so we continue the loop and if not we insert the trail $(v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)}) \hat{v}^{(v_i)} (v_i^{(\hat{v}_i)}, \hat{v}_i^{(v_i)}) v_i^{(\hat{v}_i)}$ into U at this position. This procedure is illustrated in fig. 6b. Now U is Eulerian and hence a SOET. This completes the proof. □

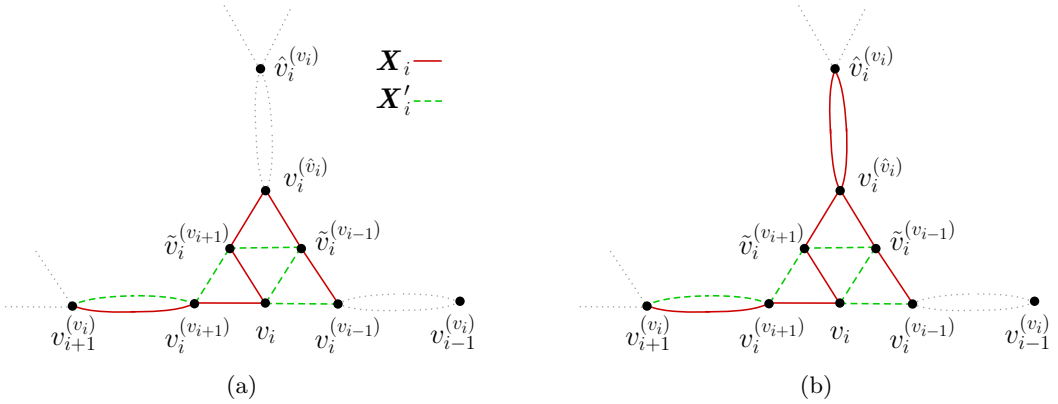


Figure 6: The trails used in the proof of lemma 3.3 to construct a SOET on a triangular-expanded graph $\Lambda(R)$ from a Hamiltonian cycle on R. Figure 6a shows the trails described by the words $\boldsymbol{X}_i$ (solid red) and $\boldsymbol{X}'_i$ (dashed green), defined in eqs. (81) and (82). Figure 6b shows how these trails can be extended to form a Eulerian tour and therefore a SOET with respect to $V(R)$. The dotted gray lines show edges of $\Lambda(R)$ which are not used by the trails.

Next we define a special type of SOET on triangular-expanded graphs, which we call HAMSOETs. These special SOETs on a triangular-expanded graph $\Lambda(R)$, are closely related to Hamiltonian cycles on R.

Definition 3.4 (HAMSOET)**.** *Let R be a 3-regular graph and $\Lambda(R)$ its triangular-expansion. Furthermore, let U be a* SOET *on $\Lambda(R)$ with respect to $V(R)$. U is called a* HAMSOET *with respect to R if, for all vertices $u, v \in V(R)$ we have that if u and v are consecutive with respect to the* SOET *U they are also adjacent in the graph R.* ◇

Next we prove that if the triangular-expansion of a 3-regular graph R allows for a HAMSOET with respect to R then the 3-regular graph R is Hamiltonian.

Lemma 3.4. *Let R be a 3-regular graph and let $\Lambda(R)$ be its triangular-expansion. If $\Lambda(R)$ allows for a* HAMSOET *with respect to R, then R is Hamiltonian.* ◇

Proof. This follows by the definition of a HAMSOET. A HAMSOET U will induce a double occurrence word of the form

$$m(U) = \boldsymbol{X}_0 s_1 \boldsymbol{X}_1 s_2 \dots s_k \boldsymbol{X}_k s_1 \boldsymbol{X}'_1 s_2 \dots s_k \boldsymbol{X}'_k. \tag{83}$$

where $s_1, \dots s_k \in V(R)$ with $k = |V(R)|$ and where $\boldsymbol{X}_i$ and $\boldsymbol{X}'_i$ are maximal sub-words associated to $s_i, s_{i+1} \in V(R)$. Now consider the induced double occurrence word $m(U)[V(R)]$. We have

$$m(U)[V(R)] = s_1 \dots s_k s_1 \dots s_k. \tag{84}$$

Consider now the sub-word $s_1 \dots s_k$ of $m(U)[V(R)]$. This sub-word describes a Hamiltonian cycle on R. To see this, note that each vertex in $V(R)$ occurs exactly once in $s_1 s_2 \dots s_k$. Furthermore, since s_i and s_{i+1} are consecutive in U, they are adjacent in R for all $i \in [k-1]$, by definition of a HAMSOET. Finally, the same also holds for s_k and s_1. Hence the tour on R described by $s_1 \dots s_k$ visits each vertex in R exactly once and is hence a Hamiltonian cycle.

□

We would like to prove that for any 3-regular graph R the existence of a SOET on its triangular-expansion $\Lambda(R)$ implies the existence of a Hamiltonian cycle on R. So far we have proven this only when $\Lambda(R)$ allows for a HAMSOET. However, a priori not all SOETs on triangular-expansions $\Lambda(R)$ have to be HAMSOETs. The reason for this is that consecutive vertices in a SOET are not necessarily adjacent in R, since a SOET can contain true skips.

In the following lemma we will prove that consecutive vertices in a SOET are actually adjacent in R, except for two special cases. These special cases can be remedied since we show that for these cases we can always find a different SOET which is actually a HAMSOET.

The arguments proven below are understood the easiest when one reproduces the visual aids given in figs. 8 and 9 as one follows the arguments. We have the following lemma.

Lemma 3.5. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. If $\Lambda(R)$ allows for a* SOET *with respect to $V(R)$, then $\Lambda(R)$ allows for a* HAMSOET *with respect to $V(R)$.* ◇

Proof. To prove this lemma we will go through the following steps

Step 1 Note that if two vertices u, v in $V(R)$ (R being a 3-regular graph) are not adjacent in R but are consecutive in a SOET U with respect to $V(R)$ on $\Lambda(R)$, then the sub-trails of U described by the maximal sub-words associated to u and v must make a non-zero number of true skips in $\Lambda(R)$

Step 2 Argue by contradiction that this non-zero number of true skips can never be greater than one. This argument leverages lemma 3.6 which states that if a sub-trail of a SOET U with respect to $V(R)$ makes true skips at triangle subgraphs T_{w_1}, T_{w_2} for $w_1, w_2 \in V(R)$ and w_1, w_2 are adjacent in R then w_1, w_2 must actually be consecutive in the SOET U and lemma 3.7 which, in a slightly different initial situation than lemma 3.6, also concludes that two vertices must be consecutive in a SOET U with respect to $V(R)$.

Step 3 Argue by contradiction that there are only two possible ways for the sub-trails described by the maximal sub-words associated to u and v to make one true skip each. This argument also leverages lemmas 3.6 and 3.7.

Step 4 Argue that if a triangular-expansion of a 3-regular graph R allows for a SOET U as in **Step 3**, then it also allows for a SOET U' that is a HAMSOET.

Details of step 1

Let R be such that $\Lambda(R)$ allows for a SOET with respect to $V(R)$. Let U be any such SOET. If U is also a HAMSOET then we are directly done, therefore assume that U is not a HAMSOET. Since U is not a HAMSOET there must exist at least two vertices $u, v \in V(R)$ which are consecutive in U, but not adjacent in R. By definition, since u and v are consecutive and U is a SOET, there must exist exactly two different maximal sub-words $\boldsymbol{X}, \boldsymbol{X'}$ of $m(U)$, associated to u and v. Since u and v are not adjacent in R, the trails described by $\boldsymbol{X}$, $\boldsymbol{X'}$ must each traverse vertices in at least one (but possibly more) triangle subgraph that is not T_u or T_v. Since $\boldsymbol{X}$ and $\boldsymbol{X'}$ are maximal sub-words, they can not contain any vertex in $V(R)$ as a letter and hence the sub-trails described by $\boldsymbol{X}, \boldsymbol{X'}$ must each make true skips (see definition 3.3) at at least one triangle subgraph.

Assume w.l.o.g. that the sub-trail described by $\boldsymbol{X}$ makes true skips at the triangle subgraphs $T_{s_1}, T_{s_2}, \dots T_{s_r}$, in this order, and similarly for $\boldsymbol{X'}$ and $T_{s'_1}, \dots T_{s'_{r'}}$. The true skips that the sub-trail described by $\boldsymbol{X}$ makes, must be at different triangle subgraphs than the ones for $\boldsymbol{X'}$, since there can only be exactly one skip per triangle subgraph, due to lemma 3.2. The triangle subgraphs $T_{s_1}, T_{s_2}, \dots T_{s_r}, T_{s'_1}, \dots T_{s'_{r'}}$ are therefore pairwise different. We will call this situation a rr'-skip, which is illustrated in fig. 7. Note that by assumption, r and r' are both greater than zero. We will first show that the case where either r or r' are greater than one can never occur if U is a SOET.

Figure 7: This figure is a visual aid for Step 1 of lemma 3.5. A simplified visualization of a triangular-expansion $\Gamma(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and vertices in $V(R)$ (diamonds) shown. Also shown are the sub-trails, of a SOET U, described by maximal sub-words X and X' associated to vertices u, v making true skips at triangle subgraphs $T_{s_1}, \ldots, T_{s_r}$ and $T_{s'_1}, \ldots, T_{s'_r}$ respectively. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Gamma(R)$.

Details of step 2

As a visual aid for the following argument, refer to fig. 8. We will make an argument by contradiction. Therefore let r be strictly greater than one. Consider the sub-trail described by the sub-word X defined above. By assumption this sub-trail makes true skips at at least two adjacent triangle subgraphs. Take these to be T_{s_1} and T_{s_2}. As we will prove in lemma 3.6, if two adjacent triangle subgraphs T_{s_1}, T_{s_2} contain true skips, then s_1 and s_2 must be consecutive in the SOET U and that there is some maximal sub-word Y of $m(U)$ associated to s_1, s_2 that contains the sub-word $s_1^{(s_2)} s_2^{(s_1)}$.

This implies that (by definition of SOET) there is some other maximal sub-word Y' of $m(U)$ associated to s_1 and s_2. This sub-word Y' cannot contain the sub-word $s_1^{(s_2)} s_2^{(s_1)}$, as $s_1^{(s_2)} s_2^{(s_1)}$ already appears in both the maximal sub-word X and the maximal sub-word Y.

This implies the sub-trail described by Y' must make a nonzero number of true skips at triangle subgraphs $T_{\hat{s}_1}, \ldots, T_{\hat{s}_k}$ in that order. Now consider the triangle subgraphs T_{s_1} and $T_{\hat{s}_1}$. As we will prove below, lemma 3.7 applied to the triangle subgraphs $T_{s_1}, T_{\hat{s}_1}$ implies that the vertices s_1 and $\hat{s}_1$ must be consecutive in U. Call the maximal sub-word of $m(U)$ associated to these vertices W_0. Moreover, by lemma 3.7 this maximal sub-word contains the sub-word $s_1^{(\hat{s}_1)} \hat{s}_1^{(s_1)}$. Similarly we can see that the vertices s_2 and $\hat{s}_k$ must be consecutive in U. Call the maximal sub-word of $m(U)$ associated to these vertices W_k. By lemma 3.7 this maximal sub-word contains the sub-words $s_2^{(\hat{s}_{k+1})} \hat{s}_{k+1}^{(s_2)}$.

By applying lemma 3.6 again to all triangle subgraph pairs $T_{\hat{s}_i}, T_{\hat{s}_{i+1}}$ for $i \in [k-1]$ we come to the conclusion that $\hat{s}_i, \hat{s}_{i+1}$ for $i \in [k-1]$ must be consecutive in U. Call the maximal sub-word of $m(U)$ associated to these vertices W_i. By lemma 3.7 these maximal sub-words contain the sub-words $\hat{s}_i^{(\hat{s}_{i+1})} \hat{s}_{i+1}^{(\hat{s}_i)}$. This implies that U must traverse the vertices $s_1, \hat{s}_1, \ldots, \hat{s}_k, s_2, s_1, \hat{s}_1, \ldots, \hat{s}_k, s_2$ in order. Since by construction $\{s_1, s_2, \hat{s}_1, \ldots, \hat{s}_k\} \in V(R)$ we have that U is not a valid SOET on $V(R)$ (this can be seen by noting that e.g. $T_{s'_1}$ already contains a true skip, hence s'_1 can not be

part of the set). Hence by contradiction we must have $r \leq 1$. We can make the same argument for r' yielding $r' \leq 1$.

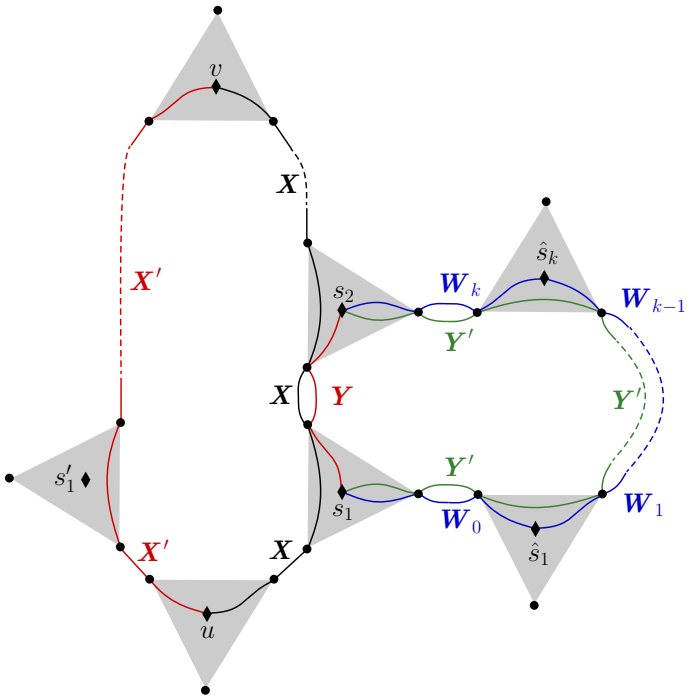


Figure 8: This figure is a visual aid for Step 2 of lemma 3.5. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and the vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. In this argument a contradiction is arrived at by first assuming the that the SOET U has a sub-trail described by the maximal sub-word $\boldsymbol{X}$ associated to the vertices u, v makes true skips at triangle subgraphs T_{s_1}, T_{s_2}. Using lemma 3.6 it is shown that the maximal sub-word $\boldsymbol{Y}$ must exist (associated to vertices s_1, s_2). This then means that the maximal sub-word $\boldsymbol{Y'}$ must exist (also associated to s_1, s_2). The sub-trail described by this sub-word must make true skips at triangle subgraphs $T_{w_1}, \ldots, T_{w_k}$. Using lemma 3.6 and lemma 3.7 it is then concluded that the SOET U must visit the vertices $s_1, w_1, \ldots w_k, s_2, s_1$ consecutively which means U is not a valid SOET (since $\{s_1, w_1, \ldots, w_k, s_2\} \neq V(R)$). This is a contradiction.

Details of step 3

As a visual aid for the following argument, refer to fig. 9. Now let $r = r' = 1$. This means the sub-trails described by the maximal sub-words $\boldsymbol{X}$ and $\boldsymbol{X'}$ associated to the vertices u and v make exactly one true skip each at triangle subgraphs triangle subgraphs $T_{s_1}, T_{s'_1}$ respectively. We will now argue that there are essentially only two ways that a SOET U with the above properties can exist on $\Lambda(R)$. This argument will again go in steps:

Step 3.1 Argue that the SOET U must have a maximal sub-word associated to the vertex s_1 and some other vertex x that describes a trail traversing the edge connecting the triangle subgraphs T_u and T_{s_1}.

Step 3.2 Argue that this sub-trail must make a true skip at the triangle subgraph T_u. Note that this is equivalent to arguing $x \neq u$.

Step 3.3 Apply the same sequence of arguments for the vertices s_1 and v and also for the vertices s'_1 and u and s'_1 and v.

Step 3.4 Conclude from the fact that the SOET U can only make a single true skip at the triangle subgraphs T_u and T_v that s_1 and s'_1 must be consecutive in U and moreover that the maximal sub-words $\boldsymbol{Z}, \boldsymbol{Z'}$ associated to s_1 and s'_2 must make true skips at T_u and T_v respectively.

Details of step 3.1

Consider the second edge connecting the triangle subgraphs T_u and T_{s_1}. This is the edge $(u^{(s_1)}, s_1^{(u)})$. Since the SOET U is Eulerian it must traverse this edge. Note also that T_{s_1} already contains a true skip (made by the sub-trail described by $\boldsymbol{X}$). This means, by lemma 3.2 that any sub-trail of U crossing the edge $(u^{(s_1)}, s_1^{(u)})$ connecting T_u, T_{s_1} must traverse the vertex s_1. Let $\boldsymbol{Z}$ be the maximal sub-word describing the sub-trail starting at s_1 and containing the sub-word $u^{(s_1)}s_1^{(u)}$. This maximal sub-word is (by definition) associated to two vertices in $V(R)$. One of these vertices is s_1 and will label the other one x.

We will now argue that $x \neq u$ and hence that the sub-trail described by $\boldsymbol{Z}$ makes a true skip at T_u. We do this by contradiction in the following argument.

Details of step 3.2

For this part of the argument assume that $x = u$. This means that u is consecutive to both s_1 and v. This means there must be some other maximal sub-word $\boldsymbol{Z'}$ associated to s_1 and u. Note that this sub-word cannot contain the sub-word $u^{(s_1)}s_1^{(u)}$ as it is already contained in the maximal sub-words $\boldsymbol{Z}$ and $\boldsymbol{X'}$.

Now consider the unique third vertex that is adjacent to u in R (the vertex adjacent to u which is not s_1 or s_1'). Let us label this vertex w.

Since $T_{s_1'}$ already contains a true skip (which implies $\boldsymbol{Z'}$ cannot connect to u by making a true skip at $T_{s_1'}$) the sub-trail described by the maximal sub-word $\boldsymbol{Z'}$ must make a true skip at w. Now consider the maximal sub-word $\boldsymbol{Y}$ of $m(U)$ that that describes a sub-trail traversing the unused edge between $T_{s_1'}$ and T_u, i.e. $\boldsymbol{Y}$ contains the sub-word $u^{(s_1')}s'^{(u)}_1$. This sub-trail must be associated to s_1' (since $T_{s_1'}$ already contains a true skip) and can not not be associated to u since u is already consecutive with two vertices in $V(R)$. This means the sub-trail described by the maximal sub-word $\boldsymbol{Y}$ must make a true skip at the triangle subgraph T_u. Since the sub-word $u^{(s_1)}s_1^{(u)}$ is already contained in $\boldsymbol{Z}$ and $\boldsymbol{X'}$ the maximal sub-word $\boldsymbol{Y}$ must contain the sub-word $u^{(w)}w^{(u)}$. Since T_w already contains a true skip this implies that w must be associated to the maximal sub-word $\boldsymbol{Y}$ and hence that w and s_1' are consecutive. This means there must be a second maximal sub-word $\boldsymbol{Y'}$ associated to w and s_1'.

Since both edges connecting T_u and T_w have already been traversed by, the sub-trail described by the maximal sub-word $\boldsymbol{Y'}$ must make true skips on some triangle subgraphs $T_{\hat{s}_1}, \ldots, T_{\hat{s}_k}$.

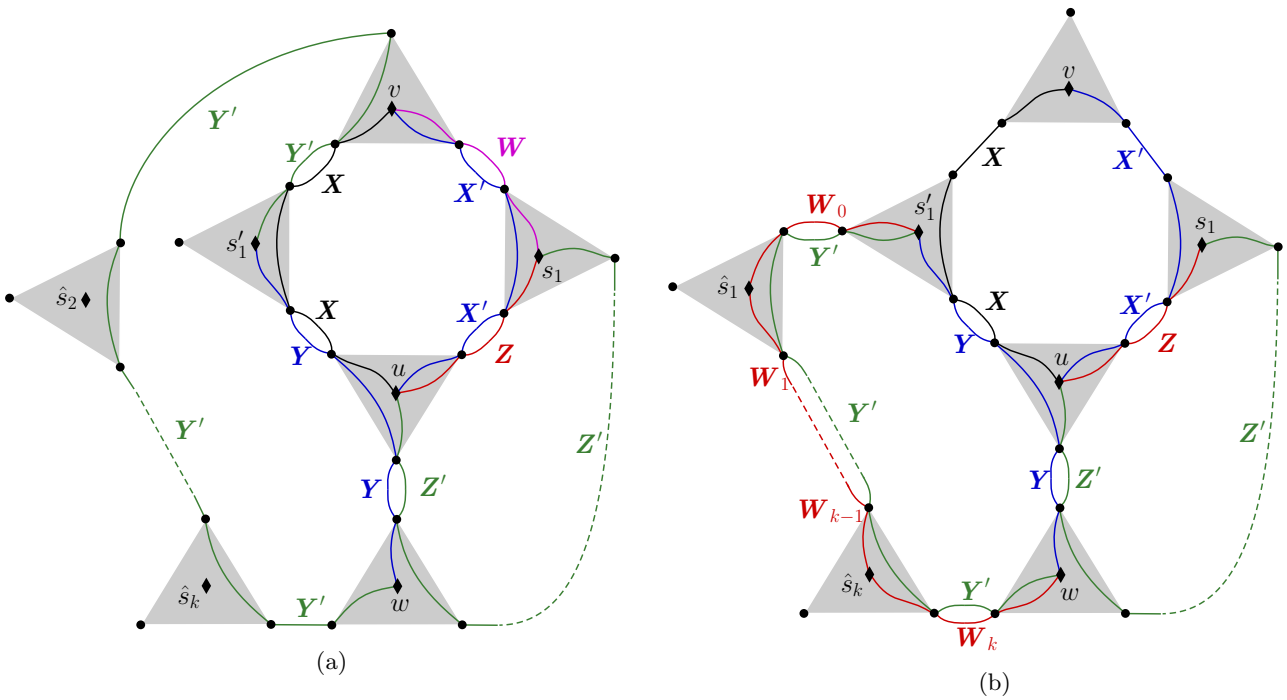


Figure 9: This figure is a visual aid for Steps 3.1 and 3.2 of lemma 3.5. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. In Step 3.2 of lemma 3.5 it is argued that if the subtrail described by the maximal sub-word $\boldsymbol{X'}$ associated to vertices u, v makes a true skip at the triangle subgraph T_{s_1} then the sub-trail described by the maximal sub-word $\boldsymbol{Z}$ associated to s_1 and another vertex x must make a true skip at the triangle subgraph T_u (or equivalently that $x \neq u$). This is done by contradiction so it is assumed (as seen in both (a) and (b)) that $x = u$. This implies the existence of the maximal sub-word $\boldsymbol{Z'}$ which must make a true skip at T_w where w is the unique neighbor of u s.t. $w \neq s_1, s_1'$. This in turn implies that s_1' and w must be consecutive and connected by a sub-trail described by the maximal sub-word labeled $\boldsymbol{Y}$ in both (a) and (b). This means another sub-trail connecting s_1' and w must exist, which described by a maximal sub-word $\boldsymbol{Y'}$. This sub-word makes true skips at triangle subgraphs $\hat{s}_1, \ldots \hat{s}_k$. There are now 2 options. Either, as shown in (a) we have that $v = \hat{s}_1$ or we have, as shown in (b) that $v \neq \hat{s}_1$. In the first case (a) lemma 3.7 is applied to conclude that v must be consecutive to s_1, s_1' and u leading to a contradiction. In the second case (b) lemma 3.6 and lemma 3.7 are used to show that any SOET U must traverse the vertices $s_1', w, \hat{s}_k, \ldots, \hat{s}_1, s_1'$ in order leading to a contradiction.

There are now two possibilities. Either (1) we have that $\hat{s}_1 = v$ (see fig. 9a) or (2) that $\hat{s}_1 \neq v$ (see fig. 9b). We will now consider both of these cases:

1. If $\hat{s}_1 = v$ we can apply lemma 3.7 to the vertices v and s_1 to conclude that $\hat{s}_1 = v$ implies that v and s_1 are consecutive. Call the maximal sub-word connecting them $\boldsymbol{W}$. We now have that the vertices u, v and s_1 are pairwise consecutive to each other. Since $\{u, v, s_1\}$ is a strict subset of $V(R)$, U cannot be a a SOET which is a contradiction.

2. Now assume that $v \neq \hat{s}_1$. By lemma 3.7 we can now conclude that w and $\hat{s}_k$ must be consecutive and that s_1' and $\hat{s}_1 \neq v$ must be consecutive. We have to perform one last construction to prove the lemma. This construction is visualized in fig. 9b. Call the maximal sub-words associated to these vertex pairs w and $\hat{s}_k$ and s_1' and $\hat{s}_1$ $\boldsymbol{W}_k$ and $\boldsymbol{W}_0$ respectively. We can again use lemma 3.6 to conclude that $\hat{s}_i$ is consecutive to $\hat{s}_{i+1}$ for all $i \in [k-1]$. Call the maximal sub-words associated to the vertices $\hat{s}_i$ and $\hat{s}_{i+1}$

$\boldsymbol{W}_i$. This implies that U must traverse the vertices $s_1', \hat{s}_1, \ldots, \hat{s}_k, w, s_1', \hat{s}_1, \ldots, \hat{s}_k, w$ in order. Since by construction $\{s_1', s_w, \hat{s}_1, \ldots, \hat{s_k}\} \neq V(R)$ we have that U is not a valid SOET on $V(R)$ (this can be seen by noting that e.g. T_u already contains a true skip, hence u can not be part of the set).

Hence in both cases we arrive at a contradiction. This means by contradiction that $x \neq u$ and thus that the sub-trail described by $\boldsymbol{Z}$ makes a true skip at T_u.

Details of step 3.3

Now similarly to Step 3.1 consider the edge connecting T_{s_1} and T_v. We can again argue that there must exist a maximal sub-word $\boldsymbol{Z}'$ associated to the vertex s_1 and some other vertex x' that describes a sub-trail that traverses this edge. By the same argument as Step 3.2 we can conclude that $x' \neq v$ and thus that $\boldsymbol{Z}'$ describes a sub-trail making a true skip at T_v.

We can make the same argument for the vertex s_1' establishing the existence of maximal sub-words $\hat{Z}, \hat{Z}'$ that describe sub-trails making true skips at T_u and T_v respectively.

Details of step 3.4

Note that the sub-trails described by $\boldsymbol{Z}$ and $\hat{\boldsymbol{Z}}$ make true skips at the triangle subgraph T_u. Since T_u, by lemma 3.2 can only contain a single true skip and since $\boldsymbol{Z}, \hat{\boldsymbol{Z}}$ are maximal sub-words we must have that $\boldsymbol{Z} \sim \hat{\boldsymbol{Z}}$ and thus that the vertices s_1 and s_1' are consecutive with respect to the SOET U. However since we must also conclude that $\boldsymbol{Z}' \sim \hat{\boldsymbol{Z}}'$ we have that s_1 and s_1' are consecutive and have two maximal sub-words associated to them. Since there are no further constraints on U imposed by the the fact that the sub-words $\boldsymbol{X}, \boldsymbol{X}'$ associated to the vertices v, u make true skips. Therefore SOETs with this type of behavior are in fact allowed. These SOETs can also be found explicitly, as can be sen in fig. 10. If a SOET U has this type of behavior (u and v are not adjacent in R but are consecutive in the SOET U on $\Lambda(R)$) we say that the SOET U has a *valid* 11-skip. Next we show that if a SOET U has a valid 11-skip, and thus is not a HAMSOET, it can always be turned into a HAMSOET by applying a fixed set of local complementation.

Details of step 4

We now show that a SOET with valid 11-skips can be turned into a HAMSOET. The two possibilities for a SOET with a valid 11-skip are shown in fig. 10. Note that these possibilities have the same 'local' structure, the only difference is how the rest of the SOET U is connected to the valid 11-skip. We first show the procedure that should be applied if the 11-skip is of the form in fig. 10a.

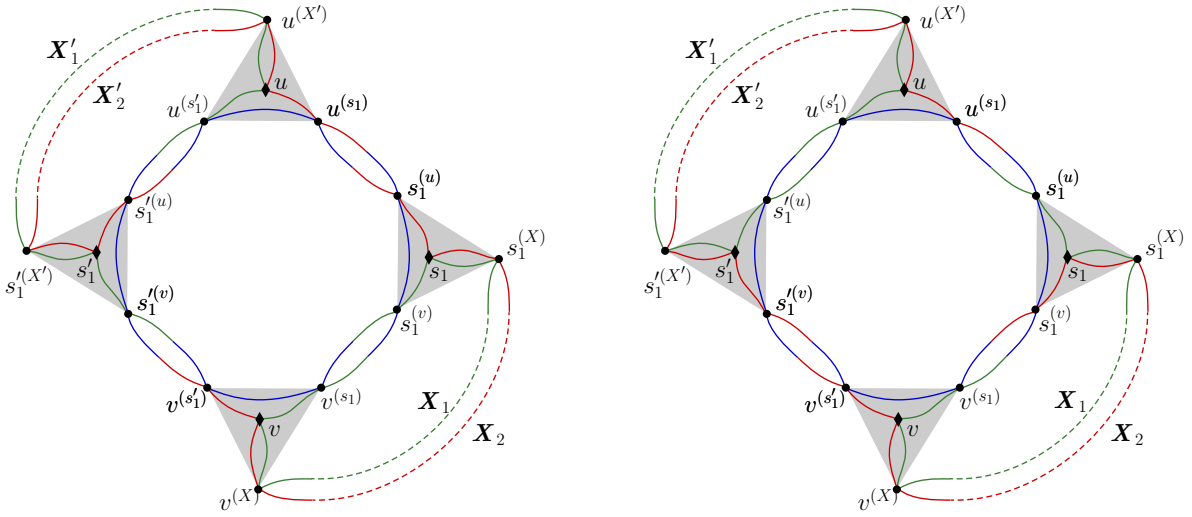


Figure 10: This figure is a visual aid for Step 4 of lemma 3.5. A simplified visualization of a triangular-expansion $\Lambda(R)$ of a 3-regular graph R is used. In the figure triangle subgraphs are shown in gray with only their outer vertices (circles) and the vertices in $V(R)$ (diamonds) shown. Also shown are sub-trail of a SOET U labeled by the maximal sub-words that describe them. These sub-trails always begin and end at a vertex in $V(R)$ but their path inside triangle subgraphs is not shown explicitly. Dashed lines are used to indicate that the sub-trails also traverse unspecified further parts of the graph $\Lambda(R)$. Figures 10a and 10b show the two valid 11-skips. The sub-word $\boldsymbol{X}_1$ describes some sub-trail of U connecting T_v and T_{s_1}. With a slight abuse of notation we denote the endpoints of this trail by $v^{(X)}$ and $s_1^{(X)}$. Similarly for $\boldsymbol{X}_2$, $\boldsymbol{X}'_1$ and $\boldsymbol{X}'_2$.

The SOET U in fig. 10a is of the form

$$m(U) = \boldsymbol{U}_1 v^{(s_1)} \boldsymbol{U}_2 v^{(s_1)} \boldsymbol{U}_3 s_1'^{(u)} \boldsymbol{U}_4 s_1'^{(u)} \boldsymbol{U}_5 \tag{85}$$

where

$$\boldsymbol{U}_1 = u u^{(s_1)} s_1^{(u)} s_1^{(v)} \tag{86}$$

$$\boldsymbol{U}_2 = v v^{(X)} \boldsymbol{X_1} s_1^{(X)} s_1 s_1^{(v)} \tag{87}$$

$$\boldsymbol{U}_3 = v^{(s'_1)} s_1'^{(v)} s'_1 s_1'^{(X')} \boldsymbol{X'_1} u^{(X')} u u^{(s'_1)} \tag{88}$$

$$\boldsymbol{U}_4 = s_1'^{(v)} v^{(s'_1)} v v^{(X)} \boldsymbol{X}_2 s_1^{(X)} s_1 s_1^{(u)} u^{(s_1)} u^{(s'_1)} \tag{89}$$

$$\boldsymbol{U}_5 = s'_1 s_1'^{(X')} \boldsymbol{X'_2} u^{(X')} \tag{90}$$

where $\boldsymbol{X_1}$ is the word associated to the sub-trail connecting $v^{(X)}$ and $s_1^{(X)}$ as seen in fig. 10a and similarly for $\boldsymbol{X'_1}, \boldsymbol{X_2}, \boldsymbol{X'_2}$ Since our goal is to make this a HAMSOET we need to have that pairs of vertices in $V(R)$ are only consecutive w.r.t. U if they are adjacent in R. This can be done by applying $\bar{\tau}$-operations to U at $v^{(s_1)}$ and $s_1'^{(u)}$. The Eulerian tour U' after these operations will be described by

$$m(U') = m\left(\bar{\tau}_{\left(v^{(s_1)}, s_1'^{(u)}\right)}(U)\right) = \boldsymbol{U}_1 v^{(s_1)} \widetilde{\boldsymbol{U}_2} v^{(s_1)} \boldsymbol{U}_3 s_1'^{(u)} \widetilde{\boldsymbol{U}_4} s_1'^{(u)} \boldsymbol{U}_5 \tag{91}$$

where the over-set tilde indicates the mirror-inverting of a sub-word. Note that neither (u, v) or (s_1, s'_1) are consecutive anymore, but instead (u, s_1) and (v, s'_1) are now consecutive w.r.t. U'. To make sure that this procedure works we need to check two things: (1) U' is a SOET and (2) there are no additional consecutive pairs of vertices in U' that are not adjacent in R. To do this, lets look at the order U and U' traverse the vertices in $V(R)$, i.e. we will look at $m(U)[V(R)]$ and $m(U')[V(R)]$. Since U is a SOET we must have that $\boldsymbol{X}_1[V(R)] = \boldsymbol{X}_2[V(R)]$ and similarly $\boldsymbol{X'}_1[V(R)] = \boldsymbol{X}'_2[V(R)]$. Lets denote these words by $\boldsymbol{X}_V = \boldsymbol{X}_1[V(R)]$ and $\boldsymbol{X}'_V = \boldsymbol{X'}_1[V(R)]$. We then have that the

double occurrence word of $m(U)$ induced by $V(R)$ is

$$m(U)[V(R)] = uv\boldsymbol{X}_V s_1 s_1' \boldsymbol{X}_V' uv \boldsymbol{X}_V s_1 s_1' \boldsymbol{X}_V' \tag{92}$$

and similarly for U' we have

$$m(U')[V(R)] = us_1 \widetilde{\boldsymbol{X}_V} vs_1' \boldsymbol{X}_V' us_1 \widetilde{\boldsymbol{X}_V} vs_1' \boldsymbol{X}_V' \tag{93}$$

It is therefore clear that the Eulerian tour U' is a SOET. Furthermore the only consecutive pairs of vertices in U' which where not consecutive in U are (u, s_1) and (v, s_1'). Since (u, s_1) and (v, s_1') are edges of R we see that we can iteratively apply this procedure to any valid 11-skip as in fig. 10a and turn the SOET into a HAMSOET. Similarly the SOET in fig. 10b can be turned into a HAMSOET by applying τ-operations to the vertices $s_1^{(u)}$ and $v^{(s_1')}$. One can explicitly check this by applying the operations to U in fig. 10b which is given by

$$m(U) = \boldsymbol{U_1} s_1^{(u)} \boldsymbol{U_2} s_1^{(u)} \boldsymbol{U_3} v^{(s_1')} \boldsymbol{U_4} v^{(s_1')} \boldsymbol{U_5} \tag{94}$$

where

$$\boldsymbol{U}_1 = uu^{(s_1)} \tag{95}$$

$$\boldsymbol{U}_2 = s_1^{(v)} v^{(s_1)} v v^{(X)} \boldsymbol{X}_1 s_1^{(X)} s_1 \tag{96}$$

$$\boldsymbol{U}_3 = u^{(s_1)} u^{(s_1')} s_1'^{(u)} s_1' s_1'^{(X')} \boldsymbol{X}_1' u^{(X')} u u^{(s_1')} s_1'^{(u)} s_1'^{(v)} \tag{97}$$

$$\boldsymbol{U}_4 = v v^{(X)} \boldsymbol{X}_2 s_1^{(X)} s_1 s_1^{(v)} v^{(s_1)} \tag{98}$$

$$\boldsymbol{U}_5 = s_1'^{(v)} s_1' s_1'^{(X')} \boldsymbol{X}_2' \tag{99}$$

with everything defined similarly to the case of fig. 10b. Going through a similar argument as above we can show that we can also turn the SOET U into a HAMSOET. This completes the lemma. □

Lemma 3.6. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. Also let u, v be adjacent vertices on R. Let U be a* SOET *on $\Lambda(R)$ with respect to $V(R)$. Let $\boldsymbol{X}$ be a maximal sub-word of $m(U)$ not associated to u and/or v containing $u^{(v)}v^{(u)}$ and describing a sub-trail that makes true skips at T_u and T_v. Then u and v are consecutive in U and moreover $m(U)$ contains a sub-word of the form*

$$u\boldsymbol{Z}_1 u^{(v)} v^{(u)} \boldsymbol{Z}_2 v, \qquad \boldsymbol{Z}_1 \subset V(T_u),\ \boldsymbol{Z}_2 \subset V(T_v), \tag{100}$$

◇

Proof. The situation described in the lemma is described graphically in fig. 11. Because U is a SOET the sub-word $v^{(u)}u^{(v)}$ must be contained exactly twice in $m(U)$. Note that at most one of these instances can be contained in the maximal sub-word $\boldsymbol{X}$. The other instance must be contained in a different maximal sub-word Z. This maximal sub-word will be associated to two vertices w_1, w_2. Note that since $v^{(u)}u^{(v)} \in Z$ and $v^{(u)}u^{(v)} \in X$ either we must have that $w_1 = u$ or that the sub-trail described by the maximal sub-word $\boldsymbol{Z}$ makes a true skip at T_u. Since T_u already contains a true skip (made by the sub-trail described by $\boldsymbol{X}$) we must have that $w_1 = u$. We can make the same argument for the vertex v. This means the maximal sub-word $\boldsymbol{Z}$ must be associated to u and v and hence that u, v must be consecutive in U and moreover we have that

$$\boldsymbol{Z} = \boldsymbol{Z}_1 u^{(v)} v^{(u)} \boldsymbol{Z}_2, \qquad \boldsymbol{Z}_1 \subset V(T_u) \setminus \{u\},\ \boldsymbol{Z}_2 \subset V(T_v) \setminus \{v\}. \tag{101}$$

□

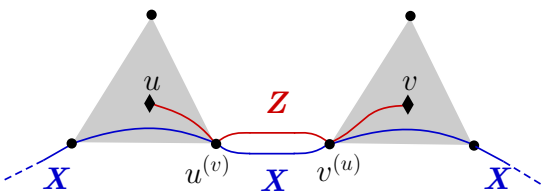


Figure 11: This figure is a graphical aid for lemma 3.6. Shown are two triangle subgraphs T_u, T_v (gray triangles) with outer vertices (circles) and the vertices u and u (diamonds) shown. The lemma starts from assuming the existence of the sub-trail described by the maximal sub-word $\boldsymbol{X}$, labeled as such in the figure. From this starting point the existence of the maximal sub-word $\boldsymbol{Z}$ associated to the vertices u and v is derived.

Lemma 3.7. *Let R be a 3-regular graph and $\Lambda(R)$ be its triangular-expansion. Also let u, v be adjacent vertices on R. Also take x_1, x_2 to be the vertices adjacent to u in R such that $x_1 \neq v, x_2 \neq v$. Let U be a SOET on $\Lambda(R)$ with respect to $V(R)$. Let $\boldsymbol{Y}$ be a maximal sub-word of $m(U)$ not associated to u and/or v containing $u^{(x_1)}x_1^{(u)}$ and $u^{(x_2)}x_2^{(u)}$ and describing a sub-trail making a true skip at T_u. Also let $\boldsymbol{X}$ be a maximal sub-word associated to u and a vertex $x_3 \neq v$ that describes a sub-trail making a true skip at v. Then u, v are consecutive and moreover $m(U)$ contains a sub-word of the form*

$$u\boldsymbol{Z}_1 u^{(v)} v^{(u)} \boldsymbol{Z}_2 v, \qquad \boldsymbol{Z}_1 \subset V(T_u),\ \boldsymbol{Z}_2 \subset V(T_v), \tag{102}$$

◇

Proof. The situation described in the lemma is described graphically in fig. 12. Because U is a SOET the sub-word $v^{(u)}u^{(v)}$ must be contained exactly twice in $m(U)$. Note that at most one of these instances can be contained in the maximal sub-word $\boldsymbol{Y}$ and none can be contained in the maximal sub-word $\boldsymbol{X}$. This means there must be a maximal sub-word $\boldsymbol{Z}$ of $m(U)$ (different from $\boldsymbol{X}$ and $\boldsymbol{X}$) containing $v^{(u)}u^{(v)}$. This maximal sub-word must again be associated with two vertices $x, \hat{x}$. If these vertices are not u, v then the sub-trail described by $\boldsymbol{Z}$ must make true skips at T_u, T_v or both. Since both of these triangle subgraphs already contain true skips this is not possible and hence $\boldsymbol{Z}$ must be associated to u and v which means they are consecutive. Moreover, by construction of $\boldsymbol{Z}$ we have

$$\boldsymbol{Z} = \boldsymbol{Z}_1 u^{(v)} v^{(u)} \boldsymbol{Z}_2, \qquad \boldsymbol{Z}_1 \subset V(T_u) \setminus \{u\},\ \boldsymbol{Z}_2 \subset V(T_v) \setminus \{v\}. \tag{103}$$

□

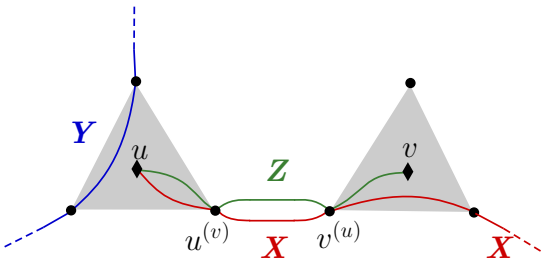


Figure 12: This figure is a graphical aid for lemma 3.7. Shown are two triangle subgraphs T_u, T_v (gray triangles) with outer vertices (circles) and the vertices u and v (diamonds) shown. The lemma starts from assuming the existence of the sub-trails described by the maximal sub-words $\boldsymbol{Y}$ and $\boldsymbol{X}$, labeled as such in the figure. From this starting point the existence of the maximal sub-word $\boldsymbol{Z}$ associated to the vertices u and v is derived.

4 Algorithms

This section is concerned with providing algorithms for various versions of the decision problems STARVERTEXMINOR with input (G, V') and VERTEXMINOR with input (G, G'). We begin by describing an efficient algorithm for STARVERTEXMINOR whenever the input graph G is distance-hereditary. We prove that this algorithm always terminates and gives correct results. We also analyze its runtime and show that it is $\mathcal{O}(|V'||V(G)|^3)$. Next we describe an algorithm for STARVERTEXMINOR whenever the input graph G is a circle graph. We prove that this algorithm is fixed-parameter tractable in the size of the input vertex-set V'. Finally we prove for the decision problem VERTEXMINOR with arbitrary connected input graphs G and G', that whenever $|V(G')| \leq 3$ and $V(G') \subset V(G)$ we have that $G' < G$ and provide an efficient algorithm for finding the sequence of local complementations and vertex-deletions that takes G to G'.

4.1 Star graph as vertex-minor of a distance-hereditary graph

In this section we present an efficient algorithm for deciding whether a star graph on a given set of vertices V' is a vertex-minor of a given distance-hereditary graph G. Throughout this section we assume that the graph G is distance-hereditary and that V' is a subset of its vertices. The algorithm presented in this section will return a sequence of vertices $\boldsymbol{v}$ in $V(G)$, such that $\tau_{\boldsymbol{v}}(G)[V'] = S_{V'}$ if such a sequence exists and raise an error-flag otherwise, indicating that

$S_{V'}$ is not a vertex-minor of G. We first present the algorithm in section 4.1.1, analyze its runtime in section 4.1.2 and prove that it is correct in section 4.1.3. An implementation in SAGE [47] of the algorithm can be found at [1].

4.1.1 The algorithm

We first give a rough sketch of the idea behind the algorithm. Remember that the task of the algorithm is to find a sequence of local complementations $\tau_{\boldsymbol{v}}$ such that the induced subgraph of $\tau_{\boldsymbol{v}}(G)$ on the vertices V' is a star graph.

The algorithm starts by choosing a one vertex c in V' which will become the center of the star graph on V'. It then proceeds by different picking vertices $v \in V'$ and making them adjacent to c by performing local complementations. After every vertex that is made adjacent the algorithm will check if the induced subgraph on the neighborhood of c is a star graph. If it is not it will attempt to turn it into a star graph by local complementations. If it fails at doing so it will raise an error and if it succeeds it will pick another vertex in V' and repeat the procedure until all vertices in V' are in the neighborhood of c. We will often call this process of making a vertex v adjacent to c 'adding' the vertex v to the star graph. To understand when the algorithm might fail we now zoom in on the situation where all but one vertex of V' has been added to the neighborhood of c. Let us call this vertex f. At this point in the algorithm the induced subgraph $G[V'\{f\}]$ is already a star graph (be previous successful iterations of this procedure).

The task is now to turn $G[V']$ into a star graph by making f adjacent to the center c of $G[V' \setminus \{f\}]$ but to no other vertex of V', and at the same time not change any edges in $G[V' \setminus \{f\}]$. This will be done in two steps, which are explained further below:

1. Make f and c adjacent, without changing any edges in $G[V' \setminus \{f\}]$. The star graph $S_{V'}$ is then a subgraph of the graph, but not necessarily an induced subgraph, since f could be also be adjacent to other vertices in V' than c. We will call these edges between f and vertices in $V' \setminus \{f\}$ *bad edges*. This first step is the task of algorithm 2 below. Interestingly, this step always succeeds if the graph is connected, even if the graph is not distance-hereditary.
2. Remove the bad edges, without changing any other edges between vertices in V'. The removal of the bad edges is the task of algorithm 1 below. Algorithm 1 tries to remove the bad edges by checking a few cases. Thus, one of the main results of this section is to prove that these cases provide a necessary condition for $S_{V'}$ being a vertex-minor of G.

We will now describe the two main steps above of the algorithm in more detail. Let's denote the vertices as above and furthermore the current leaves in $G[V' \setminus \{f\}]$ as $V' \setminus \{c, f\} = \{l_1, \ldots, l_k\}$.

Details of step 1:

The vertices c and f are made adjacent by performing local complementations along the shortest path P from c and f, see fig. 13a. The operations along the path P will in fact be either pivots, i.e. $\rho_{(u,v)} = \tau_v \circ \tau_u \circ \tau_v$, or single local complementations depending on the situation. The reason for this is to not remove edges between c and the l_i's. The details of the operations along the path P are given in algorithm 2 together with the proof in section section 4.1.3.

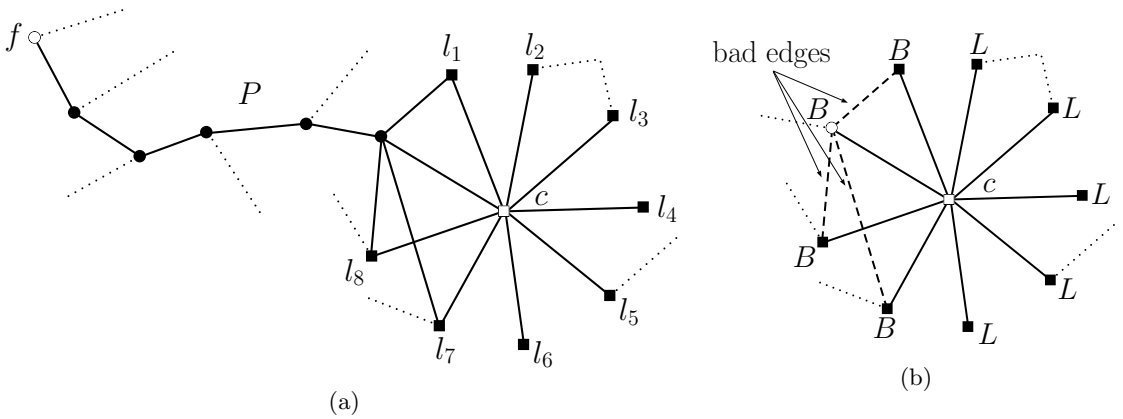


Figure 13: Visualization of how a vertex f is added to the neighborhood of c. The original graph is shown in fig. 13a where the vertices $\{c\} \cup \{l_i\}_i$ (squares) already form a star graph and the dotted lines are arbitrary edges to the rest of the graph. The new vertex f (white circle) is made adjacent to the center c (white square) by performing pivots and local complementations along the shortest path P (black circles). After the pivots and local complementations the new vertex f is adjacent to the center c of the star graph but also to some leaves B by *bad edges* (dashed lines), see fig. 13b.

As mentioned above, by making f and c adjacent we might have also added bad edges between f and some of the vertices $\{l_i\}_i$, see fig. 13b. Let's denote the set of vertices which are incident to a bad edge by B and the set of vertices not incident to a bad edge, apart from c, by $L = (V' \setminus \{c\}) \setminus B$. We call such a graph as the induced subgraph on V' a star-star graph, see definition 4.1.

Next, we must remove the bad edges in order to turn $G[V']$ into a star graph. Let G now be the graph with f and c adjacent but with possibly some bad edges.

Details of step 2:

In this step we will remove the bad edges, if we can. A situation where bad edges can be removed, as we will show, is when there exists a vertex $u \notin V'$, which is adjacent to all vertices in B but not to any vertex in L. The existence of such a vertex u is thus a sufficient condition for the removal of bad edges. When G is a distance hereditary graph, it turns out that this condition is also necessary, that is if no such vertex u exists, then the star graph on V' is not a vertex-minor of G, and we can stop the algorithm. This is shown in detail in section 4.1.3. For this statement to hold L cannot be empty, but this can always be achieved by performing a local complementation at c first if needed, which is done in line 14 in algorithm 1. Assume that there indeed exist such a vertex u, i.e.

$$(L \neq \emptyset) \ \wedge \ (u \notin V') \ \wedge \ (B \subseteq N_u) \ \wedge \ (L \cap N_u = \emptyset) \tag{104}$$

see fig. 14. Now u can be adjacent to c or not. Let's consider these cases separately:

- Case 1 u and c are not adjacent:

  Remember that f is the center of the induced star graph $G[B]$. If a local complementation is performed at u, the bad edges are removed but new ones will be created between the vertices in $B \setminus \{f\}$. These new bad edges will then form a complete graph on $B \setminus \{f\}$ and we call such a graph on the vertices $V' \setminus \{f\}$ a complete-star graph, see definition 4.2.

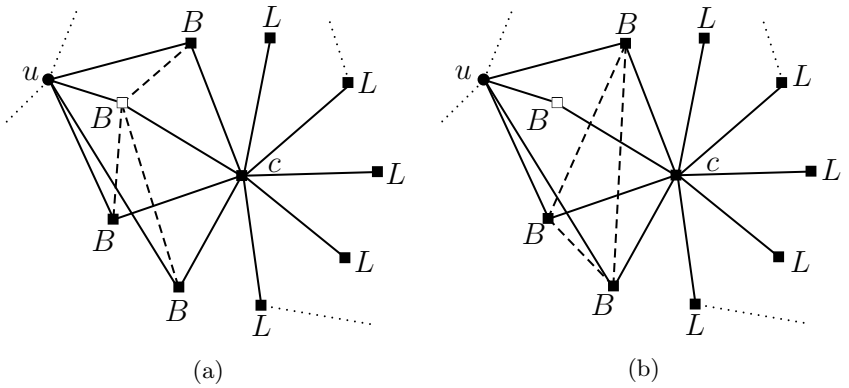


Figure 14: Visualization of how *bad edges* are removed. The original graph is shown in fig. 14a where the vertices $B \cup L \cup \{c\}$ (squares) are the desired vertices of the star graph, the dashed lines are the bad edges and vertex u (black circle) is as in eq. (105). Figure 14b shows the graph after performing a local complementation on u which produces a new leaf (white square) and makes the bad edges form a complete graph. This complete graph of bad edges can then be removed by finding a vertex u' that is adjacent to all vertices in B (and to none in L) and performing a local complementation at u'.

Performing the same step again, i.e. doing a local complementation at another vertex adjacent to all vertices in $B \setminus \{f\}$, will remove all bad edges. We have then produced the star graph on V' in two steps.

- Case 2: u and c are adjacent:

  In this case, if a local complementation is performed at u, some edges between c and vertices in L will be removed, which is not desired. We can solve this by finding another vertex h adjacent to both u and c but not to any other vertex in V', by which we can remove the edge (u, c), see fig. 15. In the following section we show that if there is no vertex h of this form, the star graph is not a vertex-minor of G and we can stop the algorithm.

To prove that the algorithm is correct we need to show that cases checked by algorithm 2 to remove the bad edges actually provides a necessary condition for $S_{V'}$ being a vertex-minor of G. To be precise, we will show that a necessary[8] condition for the star graph on V' being a vertex-minor of G is

$$\mathcal{P}(B, L, c) = \exists u \in V \setminus V' : \left(B \subseteq N_u \ \wedge \ L \cap N_u = \emptyset \ \wedge \ \left((u, c) \notin E \ \vee \ \exists h : \left(h \in N_u \cap N_c \setminus \bigcup_{x \in V' \setminus \{c\}} N_x \right) \right) \right), \quad (105)$$

where $V' = B \cup L \cup \{c\}$ and L is assumed to be nonempty. It is important to note here that this condition is only valid if the graph is in the correct form, i.e. the induced subgraph on V' form a star-star graph or a complete-star graph.

[8] This condition is not sufficient in itself, however theorem 4.1 provide a necessary and sufficient condition.

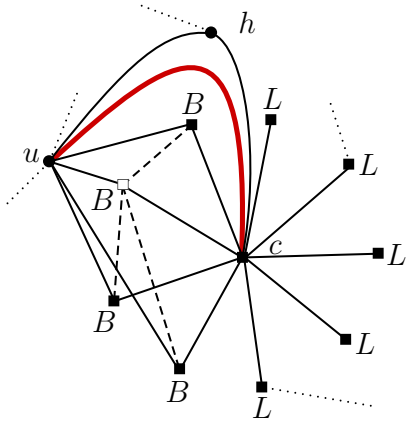


Figure 15: A visualization of the case where a vertex u used to remove the bad edges is also adjacent to c (red thick edge). The vertex h can be used to remove the edge (u, c) by applying a local complementation at h. Since h is not adjacent to any other vertex in V', no edges in the induced subgraph on V' are changed by this local complementation.

We formally state that eq. (105) is a necessary condition for the star graph on V' being a vertex-minor of G in theorem 4.1, which we prove in section 4.1.3. The theorem uses the notion of star-star and complete-star graphs which we formally define as:

Definition 4.1 (Star-star graph)**.** *A graph $G = (V, E)$ is called a star-star graph if there exists two subsets B and L and a vertex c, such that $\{B, L, \{c\}\}$ form a partition of V and $|B| > 1$. Furthermore $N_l = \{c\}$ $\forall l \in L$ and $c \in N_b$ $\forall b \in B$. Finally $G[B] = S_B$. Such a graph is denoted $SS_{(B,L,c)}$.* ◇

Definition 4.2 (Complete-star graph)**.** *A graph $G = (V, E)$ is called a complete-star graph if there exists two subsets B and L and a vertex c, such that $\{B, L, \{c\}\}$ form a partition of V and $|B| > 1$. Furthermore $N_l = \{c\}$, $\forall l \in L$ and $c \in N_b$, $\forall b \in B$. Finally $G[B] = K_B$. Such a graph is denoted $KS_{(B,L,c)}$. Note that if $|B| = 2$, G is also a star-star graph.* ◇

Theorem 4.1. *Let G be a distance-hereditary graph on the vertices V and let V' be a subset of V. Furthermore, let $V' = B \cup L \cup \{c\}$ be a partition of V' and let $S_{V'}$ be a star graph on the vertices V'. Then the following statements hold*

- *If $G[V'] = SS_{(B,L,c)}$ is a star-star graph and $|B| = 2$, then*

$$\mathcal{P}(B, L, c) \quad \Leftrightarrow \quad S_{V'} < G. \tag{106}$$

- *If $G[V'] = SS_{(B,L,c)}$ is a star-star graph then*

$$\neg\mathcal{P}(B, L, c) \quad \Rightarrow \quad S_{V'} \not< G. \tag{107}$$

- *If $G[V'] = SS_{(B,L,c)}$ is a star-star graph and f is the center of the star graph $G[B]$, then*

$$\mathcal{P}(B, L, c) \quad \Leftrightarrow \quad KS_{(B\setminus\{f\},L\cup\{f\},c)} < G. \tag{108}$$

- *If $G[V'] = KS_{(B,L,c)}$ is a complete-star graph then*

$$\mathcal{P}(B, L, c) \quad \Leftrightarrow \quad S_{V'} < G. \tag{109}$$

◇

Theorem 4.1 implicitly gives a necessary and sufficient condition for when $S_{V'}$ is a vertex-minor of G, if $G[V']$ is a star-star graph. More precisely, if the induced subgraph on V' is a star-star graph and $\mathcal{P}(B, L, c)$ is true then we know that local complementations can be performed to turn the induced subgraph on $V' \setminus \{f\}$ into a complete-star graph,

see eq. (108). Then, if $\mathcal{P}(B \setminus \{f\}, L \cup \{f\}, c)$ is again true, then a star graph can be created on V' by performing further local complementations, see eq. (109). If in any of these two steps, $\mathcal{P}(B, L, c)$ or $\mathcal{P}(B \setminus \{f\}, L \cup \{f\}, c)$ is false, then $S_{V'}$ is not a vertex-minor of G, see eqs. (107) and (109). In section 4.1.3 we prove these statements.

Algorithm 1 Producing $S_{V'}$ from a distance-hereditary graph G

INPUT: A graph G and a subset of vertices $V' \subseteq V(G)$.
OUTPUT: A sequence $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[V'] = S_{V'}$, if $S_{V'} < G$.
ERROR, if $S_{V'} \not< G$.

---

if $|V'| = 1$ **then**
  Return ()
  QUIT
end if
Find a $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)$ contain the star graph on V' as a subgraph by calling algorithm 2.
Let c be a vertex in V', adjacent to all other in V' (except itself).
for i in $\{0, 1\}$ **do** ▷ Two iterations are always needed if there is more than one bad edge
  Let B be the vertices incident to a bad edge. ▷ These are the vertices in $\tau_m(G)[V' \setminus \{c\}]$ of degree 1 or higher
  Let $L = V' \setminus (\{c\} \cup B)$.
  if $B = \emptyset$ **then** ▷ If already $S_{V'}$, only for $i = 0$
    Return $\boldsymbol{v}$
    QUIT
  else
    if $B = V' \setminus \{c\}$ **then** ▷ I.e. if $L = \emptyset$
      Set $\boldsymbol{v} = \boldsymbol{v}\|(c)$
      BREAK
    end if
    Let U be the set $U = \{u \in V(G) \setminus V' : B \subseteq N_u \wedge L \nsubseteq N_u\}$ ▷ Candidates for the u in eq. (105)
    if $U = \emptyset$ **then**
      Raise ERROR($S_{V'}$ is not a vertex-minor of G) ▷ Actually not needed, only for clarity
    end if
    Set $found$=FALSE
    for u **in** U **do**
      if $(u, c) \notin E(\tau_{\boldsymbol{v}}(G))$ **then**
        Set $\boldsymbol{v} = \boldsymbol{v}\|(u)$
        Set $found$=TRUE ▷ Found a u satisfying eq. (105)
        BREAK
      else if $\exists h : \left(h \in N_u \cap N_c \setminus \bigcup_{x \in V' \{u,c\}} N_x\right)$ **then**
        Set $\boldsymbol{v} = \boldsymbol{v}\|(h, u)$
        Set $found$=TRUE ▷ Found a u and h satisfying eq. (105)
        BREAK
      end if
    end for
    if $\neg found$ **then** ▷ I.e. condition eq. (105) is false
      Raise ERROR($S_{V'}$ is not a vertex-minor of G)
    end if
  end if
end for
Return $\boldsymbol{v}$
QUIT

Algorithm 2 Find a $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)$ contain the star graph on V' as a subgraph

1: **INPUT**: A graph G and a subset of vertices $V' \subseteq V(G)$.
2: **OUTPUT**: A sequence $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[V'] = SS_{(B,L,c)}$, where $(B, L, \{c\})$ is a partition of V'.
3: ---
4:
5: Pick an arbitrary vertex from V' and denote this f
6: Find a $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[V' \setminus \{f\}] = S_{V'}$ and denote the center c by calling algorithm 1
7: Find a shortest path $P = (p_0 = f, p_1, \ldots, p_k, p_{k+1} = c)$ between f and c.
8: **for** i in $(1, \ldots, k)$ **do**
9: **if** f is adjacent to any vertex in $V' \setminus \{c\}$ in the graph $\tau_{\boldsymbol{v}}(G)$ **then**
10: Pick an arbitrary vertex in $N_f(\tau_{\boldsymbol{v}}(G)) \cap V' \setminus \{c\}$ and denote this v
11: Set $\boldsymbol{v} = \boldsymbol{v} \| v$
12: **else**
13: Set $\boldsymbol{v} = \boldsymbol{v} \| (f, p_i, f)$
14: **end if**
15: **end for**
16: **Return** $\boldsymbol{v}$
17: QUIT

4.1.2 Runtime of the algorithm

The algorithm described in the previous section checks if a star graph with vertex set V' is a vertex-minor of a distance-hereditary graph G. Here we show that the runtime of this algorithm is $\mathcal{O}(|V'||V(G)|^3)$. We will represent subsets of a base-set as unsorted binary lists[9], where 1 indicates that an element in the base-set is in the represented set and 0 that an element in the base-set is not in the represented set. This will be the case both for sets of vertices and sets of edges. The base-set for sets of vertices will be the set of vertices $V(G)$ of the input graph G and the base-set for edges-sets will be $V(G) \times V(G)$. Thus, we assume that the input graph G is given as an unsorted binary list, of length $|V(G)|^2$, indicating which edges are in $E(G)$. This allows us to check if an edge (u, v) is in the graph or not in constant time. Furthermore, we assume that the input-set V' is also represented as an unsorted binary list, of length $|V(G)|$, indicating which of the vertices of G are in V'. We also assume that the size of $|V'|$ is given together with its representation, which allows us to faster create representations of subsets of V'.

Sets used internally by the algorithm (B, L and U) will also be represented as unsorted binary lists together with the size of the sets. The sizes of the sets will be updated accordingly whenever an element is added. Note that B and L are subsets of V' and will therefore be represented as unsorted binary lists, of length $|V'|$, indicating which elements of V' are in these sets. However, U is not a subsets of V' and will therefore be represented by an unsorted binary list of length $|V|$. Thus, given a vertex v, checking if v is in a set of vertices V can be done in constant time and adding a vertex to a set can be done in constant time (flipping the bit at the corresponding position). Furthermore, iterating over elements in a set can be done in linear time with respect to the base-set, i.e. $\mathcal{O}(|V(G)|)$ for V' and U and $\mathcal{O}(|V'|)$ for B and L.

As described, the full algorithm starts by calling algorithm 1, which in turn calls algorithm 2, which again calls algorithm 1 and so on. We will see that the computation that dominates the runtime is updating the graph $\tau_{\boldsymbol{v}}(G)$ whenever $\boldsymbol{v}$ is concatenated, as in line 13 of algorithm 1 and line 6 of algorithm 2. We will assume that both algorithm 1 and algorithm 2 have access to a common graph which they can update to $\tau_{\boldsymbol{v}}(G)$, whenever $\boldsymbol{v}$ is concatenated, to prevent this from being done for the whole sequence $\boldsymbol{v}$ every time. Note that $\tau_{\boldsymbol{v}}(G)$ takes up the same amount of space regardless of $\boldsymbol{v}$. Each local complementation in the sequence can be performed in time $\mathcal{O}(|V(G)|^2)$ [11]. Since algorithm 1 and algorithm 2 increase the length of $\boldsymbol{v}$ by $\mathcal{O}(1)$ and $\mathcal{O}(|V(G)|)$ respectively each call, the runtime to update the graph $\tau_{\boldsymbol{v}}(G)$ is $\mathcal{O}(|V(G)|^3)$. We will now show that all other parts of both algorithm 1 and algorithm 2 takes time less than $\mathcal{O}(|V(G)|^3)$, which will imply that the total runtime is $\mathcal{O}(|V'||V(G)|^3)$ since algorithm 2 is called $\mathcal{O}(|V'|)$ times[10]. Let's start by going through the runtime of algorithm 1 line by line:

- Line 6 (and 15, 24): Checking if there is only one element (or none) in V' (in B, in U) can be done in constant time, since we keep track of the sizes of these sets.

[9] It is possible to represent the sets in different ways, by for example (un)sorted lists containing the vertices as entries. However most reasonable data structures will not affect the total runtime of the algorithm but can reduce the memory used.
[10] Note that algorithm 2 calls algorithm 1 with the set $V' \setminus \{f\}$, thereby reducing the size of V' in each recursive call.

- Line 11: Finding a vertex $c \in V'$ adjacent to all vertices in V' (except itself) can be done in time $\mathcal{O}(|V'|^2)$ by for each vertex v in V' checking if $\tau_{\boldsymbol{v}}(G)$ contains all edges in the set $\{(v, w) : w \in V' \setminus \{v\}\}$. Let c be the first such vertex v.
- Line 13 and 14: Constructing the sets B and L can be done in time $\mathcal{O}(|V'|^2)$ by checking, for each vertex v in $V' \setminus \{c\}$, if $\tau_{\boldsymbol{v}}(G)$ contains at least one edge from the set $\{(v, w) : w \in V' \setminus \{c\}$. If this is the case, v will be added to the array representing B, otherwise v will be added to L.
- Line 19: Checking if $B = V' \setminus \{c\}$ can be done in time $\mathcal{O}(|V'|)$ by checking if all entries of the list representing B are 1, except at position c.
- Line 23: Constructing the set U can be done in time $\mathcal{O}(|V'||V(G)|)$ by checking, for each u in $V(G) \setminus V'$, that $\tau_{\boldsymbol{v}}(G)$ contains all edges in the set $\{(u, w) : w \in B\}$ and no edges in the set $\{(u, w) : w \in L\}$. If this is the case, u will be added to the array representing U.
- Line 28-38: The body of this *for*-loop will be executed $\mathcal{O}(|V(G)|)$ since there are at most $\mathcal{O}(|V(G)|)$ elements in U.
  - Line 29: Checking if (u, c) is an edge in $\tau_{\boldsymbol{v}}(G)$ can be done in constant time.
  - Line 33: Finding a vertex h which is adjacent to both u and c but to no other vertex in V' can be done in time $\mathcal{O}(|V'||V(G)|)$ (or determining that there is none), by first finding the neighbors of u, i.e all the vertices h such that (u, h) is an edge in $\tau_{\boldsymbol{v}}(G)$ and then, for each neighbor h of u, checking if h is also adjacent to c but to no other vertex in V'. This is done by checking if (h, c) is an edge in $\tau_{\boldsymbol{v}}(G)$ and that no element of the set $\{(h, w) : w \in V' \setminus \{c\}\}$ is.

Thus, the total runtime of algorithm 1, except for the recursive call to algorithm 2 in line 10, is $\mathcal{O}(|V'||V(G)|^2)$ (from line 33 in the *for*-loop.).

The runtime of each command in algorithm 2 is:

- Line 5: Picking the vertex f can be done in constant time (pick the first entry).
- Line 7: Finding a shortest path between f and c can be done in time $\mathcal{O}(|V(G)|^2)$ by using Dijkstra's algorithm [20].
- Line 8-15: The body of this *for*-loop will be executed $\mathcal{O}(|V(G)|)$ since the shortest path P is necessarily shorter than the number of vertices in $\tau_{\boldsymbol{v}}(G)$.
  - Line 9: Checking if f is adjacent to any vertex in $V' \setminus \{c\}$ can be done in time $\mathcal{O}(|V'|)$ by checking if any of the edges $\{(f, w) : w \in V' \setminus \{c\}$ are in $\tau_{\boldsymbol{v}}(G)$.
  - Line 10: The column with entry 1 in line 9 can be used for v here and thus only adds a constant time to the runtime.

Thus, the total runtime of algorithm 1, except for the recursive call to algorithm 2 in line 6, is $\mathcal{O}(|V(G)|^2)$ (from line 7 in the *for*-loop).

To further substantiate the efficiency of the algorithm we give actual run-times for an implementation of the algorithm in fig. 1.

4.1.3 Proof that the algorithm is correct

In this section we prove that the algorithm presented in the previous section works, i.e. it gives a sequence of local complementations $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[V'] = S_{V'}$, given a distance-hereditary graph G, if such a sequence exists. It is relatively easy to show that the algorithm gives the desired results when it does not return an error, which we show in section 4.1.3.1. The hard part is to prove that, when the algorithm gives an error-flag it is in fact not possible to produce the star graph, i.e. the star graph is not a vertex-minor of G, which is done in section 4.1.3.2. The notation will be the same as in the previous section, c is a vertex in V' and is adjacent to the rest of the vertices in V'. The vertices in $G[V' \setminus \{c\}]$ with degree greater than 0, i.e. the vertices incident on some bad edge, are denoted as the set B.

4.1.3.1 Algorithm succeeds

In this section we show that if algorithm 1 returns a sequence $\boldsymbol{v}$, i.e. does not give an error-flag, then $\tau_{\boldsymbol{v}}(G)[V'] = S_{V'}$. We start by showing that algorithm 2 always succeeds and gives the desired result, assuming that algorithm 1 works. After performing a pivot $\rho_{(v,u)}$, i.e. $\tau_u \circ \tau_v \circ \tau_u$, any neighbor of v will become a neighbor of u, except u itself. This means that after the first pivot in line 9 in algorithm 2, i.e. $\rho_{(p_1,f)}$, f and p_2 will be adjacent. We want to inductively show that this implies that after performing pivots along the whole path, f and c are adjacent. To do this we only need to make sure that a pivot does not remove edges in the later part of the path. More precisely, the pivot $\rho_{(p_i,f)}$ should not remove an edge (p_j, p_{j+1}) for $j > i$. The fact that later edges in the path are not removed follows from the properties of the pivot and that the path is a shortest path. Apart from edges incident on u or v, a pivot $\rho_{(v,u)}$ can only flip edges in the set $N_v \times N_u$. This shows that the pivot $\rho_{(p_i,f)}$ cannot remove an edge (p_j, p_{j+1}) since neither p_j or p_{j+1} is equal to f or p_i or is adjacent to f. If p_j or p_{j+1} would be adjacent to f, then this would not be a shortest path. We also need to make sure that we do not remove the edges from $E(S_{V'\setminus\{f\}}) = \{(c, v) : v \in V' \setminus \{c, f\}\}$, when doing pivots along the path. By the same argument above we have that the pivot $\rho_{(p_i,f)}$ can only remove an edge in $E(S_{V'\setminus\{f\}})$ if f is adjacent to a vertex in $V' \setminus \{c\}$. This is the reason for the if-statement in line 5 in algorithm 2, where we then just perform a local complementation on the corresponding vertex in $v \in V' \setminus \{c\}$ which will make f and c adjacent. Performing the local complementation on such a vertex v will not remove edges in $E(S_{V'\setminus\{f\}})$, since v is a leaf in the induced subgraph on V'. Note that there are only two cases where *bad edges* are created. If f and c are made adjacent by a local complementation on a vertex $v \in V' \setminus \{c\}$, as in line 7, the bad edge (f, v) will be created. On the other hand, if this is not the case but the last vertex p_k is adjacent to some vertices $U \subseteq (V' \setminus \{c, f\})$, then the bad edges $\{(f, u)\}_{u\in U}$ will be created. In both of these cases $\tau_{\boldsymbol{v}}(G)[V']$ will be a star-star graph, see definition 4.1. Note that f can also be adjacent to some vertices in $V' \setminus \{f\}$, even before we perform the local complementations, but these edges will still form a star graph with f as the center. If one wants to minimize the number of local complementations and use local complementation instead of pivots along the path, this is in fact possible. The only place where a pivot is needed instead of a local complementation is towards the end of the path, when p_i is adjacent to a vertex in $V' \setminus \{c, f\}$ not on the path.

What is left to show is that if algorithm 1 succeeds and returns a $\boldsymbol{v}$, then $\tau_{\boldsymbol{v}}(G) = S_{V'}$. This is easy to see, since if we perform local complementations on such vertices we are looking for, i.e. u and possibly h in eq. (105), we will remove the bad edges and produce the star graph. If $|B| > 2$ this has to be done twice, as captured by the loop over i in algorithm 1. The reason for this is that, when doing a local complementation on such a u we complement the induced subgraph $G[B]$. Since $G[B]$ is a star graph, the induced subgraph after the local complementation will be a complete graph plus a single disconnected vertex which was the center of $G[B]$. Performing the step once more will then complement the complete graph, without the disconnected vertex, and all bad edges are thus removed.

Note that we have nowhere in this section used the assumption that the graph is distance-hereditary. This implies that if the algorithm succeeds we know that $\tau_{\boldsymbol{v}}(G) = S_{V'}$, independently of whether G is distance-hereditary, in fact even independently of the rank-width of G. Furthermore, since algorithm 2 always succeeds to make $\tau_{\boldsymbol{v}}(G)[V']$ connected and from the fact that any connected graph on two or three vertices is either a star graph or a complete graph, this implies that a star graph on any subset of size two or three is a vertex-minor of G, if the vertices are connected in G, which we make use of in section 5. On the other hand, if the algorithm stops and gives an error-flag, then we do not know in general if $S_{V'}$ is a vertex-minor of G or not. In the next section we show that if the graph is distance-hereditary and the algorithm gives an error-flag we actually do know that $S_{V'}$ is not a vertex-minor of G.

4.1.3.2 Algorithm gives error

In this section we prove that if algorithm 1 gives an error-flag, i.e. if $\mathcal{P}(B, L, c)$ in eq. (105) is false, then the star graph is not a vertex-minor of the input graph. At the steps in the algorithm where the error-flag can be raised, we know that the induced subgraph on V' is either a star-star graph (definition 4.1) or a complete-star (definition 4.2) graph as shown in section 4.1.3.1. The proof will follow the following sequence of steps.

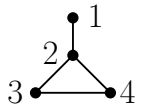
1. Prove for any distance-hereditary graph G that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a star-star graph (or complete-star graph) where $|V'| = 4$ then $S_{V'}$ is not a vertex-minor of G. This is done in theorem 4.2.
2. Use the case proven in step 1 to argue that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a star-star graph where $|V'| > 4$ then $S_{V'}$ is not a vertex-minor of G. This is done in theorem 4.3.
3. Use the case proven in step 1 to argue that if $\mathcal{P}(B, L, c)$ is false and $G[V']$ is a complete-star graph where $|V'| > 4$ then $S_{V'}$ is not a vertex-minor of G. This is done in theorem 4.5.

4.1.3.3 Proof for a star-star (complete-star) graph of size 4

We will first show in theorem 4.2 that if $\mathcal{P}(B, L, c)$ is false and $|V'| = 4$, then $S_{V'}$ is not a vertex-minor of G. This will then allows us to prove the statement for the cases where $|V'| \geq 4$.

Theorem 4.2. *Let's assume that G is a distance-hereditary graph with the following induced subgraph (which is both a star-star and a complete-star-graph)*

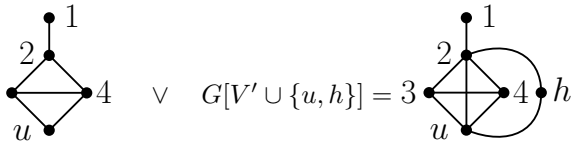
$$G[V' = \{1, 2, 3, 4\}] = \quad . \tag{110}$$



Furthermore assume that $\mathcal{P}(B, L, c)$ in eq. (105) is false, where $B = \{3, 4\}$, $L = \{1\}$ and $c = 2$. Then $S_{V'} \not\leq G$. $\diamond$

Proof. We will prove this by first showing that if $\mathcal{P}(B, L, c)$ is false and $|V(G)| > 4$, then there exist a removable leaf or twin.[11] This then implies the we can actually delete removable leaves and twins, i.e. vertices in $T(G) \setminus V'$, until there is only the vertices in V' left. Note that, if $\mathcal{P}(B, L, c)$ is false, then it is also false for any graph reached by deleting vertices in $V \setminus V'$. From theorem 2.7, i.e. the fact that deletion of removable twins or leafs does not change the property of whether a graph on a V' is a vertex-minor and that $S_{V'} \neq_{LC} G[V']$, the theorem follows. More visually, $\mathcal{P}(B, L, c)$ is false if there exist no $u, h \in V \setminus V'$ such that

$$G[V' \cup \{u\}] = \quad \vee \quad G[V' \cup \{u, h\}] = \quad . \tag{111}$$



There are only two ways for $\mathcal{P}(B, L, c)$ to be false; either $(N_3 \cap N_4) \setminus \{2\} = \emptyset$ or

$$(N_3 \cap N_4) \setminus \{2\} \neq \emptyset \quad \wedge \quad \forall u \in (N_3 \cap N_4) \setminus (N_1 \cup \{2\}) : \Big((u, 2) \in E(G) \wedge (N_u \cap N_2) \setminus \big(N_1 \cup N_3 \cup N_4\big) = \emptyset\Big). \tag{112}$$

We will consider these two cases separately and prove that if either is true, then $T(G) \setminus V' \neq \emptyset$. In most cases below we will do this by showing that one of the four vertices in V' is not in $T(G)$ which shows that $T(G) \setminus V' \neq \emptyset$ since $T(G) \geq 4$ by theorem 2.8.

Case 1:

To prove that if $(N_3 \cap N_4) \setminus \{2\} = \emptyset$ and $|V(G)| > 4$, then there exists a removable leaf or twin, we consider the following cases.

- Assume that $|(N_3 \cup N_4) \setminus \{3, 4\}| > 1$. Since, by assumption $N_3 \cap N_4 = \{2\}$, 3 and 4 does not form a twin-pair. Also, neither 3 nor 4 is not a twin, since the twin-partner would have to be a common neighbor of 3 and 4. Furthermore, neither 3 or 4 is a leaf. Thus, the only way for $3 \in T(G)$, is if 3 is an axil, requiring some vertex not in V' being a leaf.[12] Finally, if $3 \notin T(G)$, then there exist a vertex in $T(G)$ which is not in V', since by theorem 2.8 we know that $|T(G)| \geq 4$.
- Assume that $(N_3 \cup N_4) \setminus \{3, 4\} = \{2\}$.
  - Assume that $|N_1| > 1$. Then 1 is not a leaf and 2 is not an axil. Furthermore, since nothing else is connected to 3 and 4, 2 is not a twin. Thus, the only way for $2 \in T(G)$, is if 2 is an axil, requiring some vertex not in V' being a leaf. Finally, if $2 \notin T(G)$, then since $|T(G)| \geq 4$ by theorem 2.8, there must exist a vertex in $T(G)$ which is not in V'.
  - Assume that $|N_1| = 1$. Then 2 is necessarily a cut-vertex and $G \setminus 2$ will contain a connected component with no vertices in V', since $|V(G)| > 4$,. Thus, there exist a vertex in $T(G)$ which is not in V' by corollary 2.8.1.

Case 2:

To prove that if eq. (112) is true then there exists a removable leaf or twin, we consider the following cases.

[11] As in section section 2.5, removable means a vertex not belonging to the target vertices V'
[12] The same for 4.

- Assume that $|N_1| > 1$. Then if 2 is an axil, the corresponding leaf cannot be in V', since 1 is not a leaf. Furthermore, since 2 is not a leaf, if $2 \in T(G)$ then 2 has a twin-partner not in V', which is then also in $T(G)$. On the other hand if $2 \notin T(G)$, then there exist a vertex in $T(G)$ which is not in V', by theorem 2.8.
- Assume that $N_1 = \{2\}$.
  - Assume that $|(N_3 \cup N_4) \setminus \{3,4\}| > |N_3 \cap N_4|$. In this case, 3 and 4 does not form a twin-pair. Furthermore, neither 3 or 4 is a leaf. Thus the only only way for 3(or 4) $\in T(G)$, is if 3(4) is an axil or a twin, requiring some vertex not in V' being a leaf or a twin. Finally if $3(4) \notin T(G)$, then there exist a vertex in $T(G)$ which is not in V', since by theorem 2.8 we know that $|T(G)| \geq 4$.
  - Assume that $(N_3 \cup N_4) \setminus \{3,4\} = N_3 \cap N_4$. We will for this case show that $|T(G) \setminus V'| > 0$ by assuming that $T(G) = V'$ and arriving at a contradiction. Since this implies that $T(G) \neq V'$ and from the fact that $|T(G)| \geq 4$, we know that $|T(G) \setminus V'| > 0$. Consider the induced subgraph $G \setminus 4$ which is also distance-hereditary.[13] From theorem 2.8 we know that $|T(G \setminus 4)| \geq 4$, since $(N_3 \cap N_4) \setminus \{2\} \neq \emptyset$ and therefore $|G \setminus 4| \geq 4$. Thus, there is a vertex $v \notin V'$ such that $v \in T(G \setminus 4)$ but $v \notin T(G)$. Note that by the assumption from eq. (112), any neighbor of 4, except 2, is also a neighbor of both 2 and 3. The removal of 4 cannot therefore create a new leaf in $V \setminus V'$. The only option left is if there are two vertices $v, v' \in V(G) \setminus \{4\}$, such that v, v' form a twin-pair in $G \setminus 4$ but not in G. If v and v' are such vertices, it must be the case that 4 is adjacent to exactly one of v and v'. Assume without loss of generality that 4 is adjacent to v' but not to v. The neighborhoods of these vertices are then

$$N_v = N_{v'} \setminus \{4\} \quad \wedge \quad 4 \in N_{v'}. \tag{113}$$

Note that the vertices adjacent to 4 are $(N_3 \cap N_4) \cup \{3\}$. Firstly, v' cannot be in $N_3 \cap N_4$, since v is then necessarily adjacent to 3 but not to 4 which contradicts the assumption that $(N_3 \cup N_4) \setminus \{3,4\} = N_3 \cap N_4$. Secondly, v' cannot be 3, since v is then necessarily a neighbor of 2 and all vertices in $N_3 \cap N_4$, contradicting the second part of eq. (112).

□

4.1.3.4 Proof for star-star graphs

We are now able to prove the same statement as in theorem 4.2 but for $|V'| \geq 4$. The case when $G[V']$ is a star-star graph is given in theorem 4.3.

Theorem 4.3. *Let's assume that G is a distance-hereditary graph and V' is a subset $V' \subseteq V(G)$ such that the induced subgraph $G[V']$ is a star-star graph $SS_{(B',L',c')}$. Furthermore assume that $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \not< G$.* ◇

Proof. Pick an edge in $(b_1, b_2) \in G[B']$, which exist since $|B'| > 1$ and $G[B']$ is a star graph. We will prove this by first showing that

$$\neg\mathcal{P}(B', L', c') \quad \Rightarrow \quad \exists l \in L' : (\neg\mathcal{P}(\{b_1, b_2\}, \{l\}, c')).^{14} \tag{114}$$

Then from theorem 4.2 we know that $S_{\{l,c',b_1,b_2\}} \not< G$ for some $l \in L'$ and the corollary follows, because if $S_{\{l,c',b_1,b_2\}}$ is not a vertex-minor of G then neither is $S_{V'}$, since $S_{\{l,c',b_1,b_2\}} < S_{V'}$. To show that eq. (114) is true, we instead show the contrapositive statement, i.e.

$$\forall l \in L' : (\mathcal{P}(\{b_1, b_2\}, \{l\}, c')) \quad \Rightarrow \quad \mathcal{P}(B', L', c'). \tag{115}$$

Let $\mathcal{Q}(u, B, L, c)$ be the expression on $\mathcal{P}(B, L, c)$ such that

$$\mathcal{P}(B, L, c) = \exists u \in V \setminus V' : \mathcal{Q}(u, B, L, c) \tag{116}$$

For each $l \in L'$, let u_l be a vertex in $V \setminus \{l, c', b_1, b_2\}$ such that $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. We will now show that for at least one of these u_l, $u_l \in V \setminus V'$ and $\mathcal{Q}(u_l, B', L', c')$ is true. To show that for at least one u_l, $u_l \in V \setminus V'$ and $\mathcal{Q}(u_l, B', L', c')$ is true we will go trough the following steps:

1. Show that

$$\forall l \in L' : (u_l \notin V') \tag{117}$$

[13] Induced subgraphs of a distance-hereditary graph are distance-hereditary.
[14] Remember that $L' \neq \emptyset$, by definition of a star-star graph.

2. Show that
$$\forall l \in L' : (B' \subseteq N_{u_l}) \tag{118}$$

3. Show that
$$\exists l \in L' : (L' \cap N_{u_l} = \emptyset) \tag{119}$$

4. Fix $l \in L'$ to be such that the corresponding expression in eq. (119) is true.

5. Show that
$$(u_l, c') \in E(G) \ \vee \ \exists h : \left(h \in N_{u_l} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right) \tag{120}$$

If all the statements in the above steps are shown to be true we know that there exist a u_l in $V \setminus V'$ such that $\mathcal{Q}(u_l, B', L', c')$ and therefore $\mathcal{P}(B', L', c')$ is true. It is important to note that even if we consider the statements in the above steps separately we know that there exist at least one u_l that simultaneously satisfy all. To see this, note that we only use a existential quantifier in step 3 and in step 5 we consider a u_l which satisfies the corresponding property in step 3. Let's now consider the steps 1 through 5 one by one.[15] We will in these steps often claim that certain small graphs are not distance-hereditary. Verifying this can be done by hand or using our code supplied at [1].

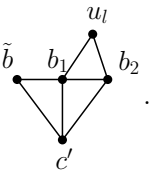
Step 1:

Firstly, since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true we know that $u_l \neq c'$. Similarly, we know that $u_l \in (N_{b_1} \cap N_{b_2}) \setminus \{c'\}$, thus u_l is not in B', since $G[B']$ is a star graph. Furthermore, since b and l are not adjacent $\forall b \in B'$ and $\forall l \in L'$, u_l is not in L'. We therefore know that $u_l \notin V'$. $\diamond$

Step 2:

To see that $B' \subseteq N_{u_l}$, assume that first that this is not the case, i.e. $\exists \tilde{b} \in B' : \tilde{b} \notin N_{u_l}$. We will now show that this contradicts the distance-hereditary property. Note that $\tilde{b}$ is not the center of the star graph $G[B]$, because either b_1 or b_2 is the center, since (b_1, b_2) is an edge in $G[B]$. Let's assume without loss of generality that b_1 is the center of $G[B]$. Furthermore, let's consider the cases where u_l and c' are adjacent or not separately.

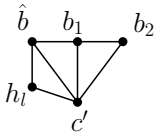
- Assume that u_l and c' are not adjacent and consider the following induced subgraph

$G[\{c', \tilde{b}, b_1, b_2, u_l\}] =$  . (121)

  The graph in eq. (121) is not distance-hereditary, since the distance between for example $\tilde{b}$ and u_l increase if b_1 is removed. This is therefore a contradiction to the assumption that G is distance-hereditary.

- Assume that u_l and c' are adjacent. We then know that there exist a vertex h_l which is adjacent to u_l and c', since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. First, let's show that h_l and $\tilde{b}$ are not adjacent. In fact we will show that h_l is not adjacent to any vertex in B, which will be useful in step 5.

  Assume the opposite, i.e. h_l is adjacent to some vertex $\hat{b} \in B' \setminus \{b_1, b_2\}$. We already know that h_l is not adjacent to b_1 or b_2, since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true. Consider therefore the following induced subgraph

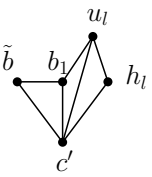
$G[\{c', \hat{b}, b_1, b_2, h_l\}] =$  (122)

  which is not distance-hereditary and we therefore know that $N_{h_l} \cap B' = \emptyset$.

[15] Step 4 is trivial.

Consider now on the other hand the following induced subgraph, with the knowledge that h_l is not adjacent to $\tilde{b}$

$$G[\{c', \tilde{b}, b_1, u_l, h_l\}] = \quad (123)$$



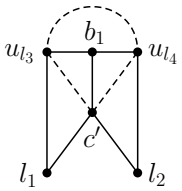
which is also not distance-hereditary.

Since in all cases we arrived at a non-distance-hereditary graph we know that $B' \subseteq N_{u_l}$. ◇

Step 3:

We show that at least for one of the u_l, $L' \cap N_{u_l} = \emptyset$. We will do this by contradiction, assume therefore that $\forall l \in L' : (L' \cap N_{u_l} \neq \emptyset)$. Since $\mathcal{Q}(u_l, \{b_1, b_2\}, \{l\}, c')$ is true, we know that $\{l\} \cap N_{u_l} = \emptyset$ for all $l \in L'$. Consider now the graph $G[L' \cup \{u_l\}_l]$. From theorem 4.4 we know that there exist $l_1, l_2 \in L'$ and $u_{l_3}, u_{l_4} \in V \setminus V'$, such that u_{l_3} is adjacent to l_1 but not to l_2 and u_{l_4} is adjacent to l_2 but not to l_1.[16] But this is in contradiction with that the graph is distance-hereditary. To see this, consider the induced subgraph

$$G[\{c', b_1, l_1, l_2, u_{l_3}, u_{l_4}\}] = \quad (124)$$

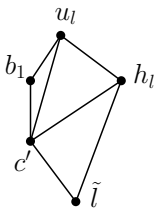


which is not distance-hereditary, independently if the edges (u_{l_3}, u_{l_4}), (c', u_{l_3}) and (c', u_{l_4}) are individually present or not. Since this contradicts the distance-hereditary property we know that $\exists l \in L' : (L' \cap N_{u_l} = \emptyset)$. ◇

Step 5:

Let's assume that u_l is then a vertex such that $B \subseteq N_{u_l}$ and $L' \cap N_{u_l} = \emptyset$. If u_l is not adjacent to c', then clearly $\mathcal{Q}(u_l, B', L', c')$. On the other hand if u_l and c' are adjacent we know that there exist a h_l in $N_{u_l} \cap N_{c'} \setminus \bigcup_{x \in \{l, b_1, b_2\}} N_x$. We thus need to show that h_l is not adjacent to any vertex in V', other than c'. Firstly, h_l cannot be adjacent to a vertex in L', since this would violate the distance-hereditary property. To see this, assume that h_l is adjacent to $\tilde{l} \in L'$ and consider the following induced subgraph

$$G[\{c', b_1, \tilde{l}, u_l, h_l\}] = \quad (125)$$



which is not distance-hereditary. This is a contradiction with the distance-hereditary property and we therefore know that $N_{h_l} \cap L' = \emptyset$. As we already shown in step 2, h_l is also not adjacent to any vertex in B'. Thus, h_l is not adjacent to any vertex in $V' = B' \cup L' \cup \{c'\}$. ◇

We have therefore shown that eq. (115) is true which implies that eq. (114) is true. Finally, as we described in the beginning of the proof, this implies that if $\mathcal{P}(B', L', c')$ is false then $S_{V'} \not\leq G$. □

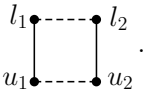
[16] Note that for example l_1 and l_3 could be the same vertex, but not necessarily.

Theorem 4.4. *Assume G is a graph on the vertices $U \cup L$ such that $U \cap L = \emptyset$ and $U \neq \emptyset$. Furthermore, assume that for each l in L, there is at least one vertex in U not adjacent to l and for each u in U, there is at least one vertex in L adjacent to u, i.e. G satisfies the following expression*

$$\mathcal{R}(U, L) = \forall l \in L : \Big(\exists u \in U : u \notin N_l\Big) \ \wedge \ \forall u \in U : \Big(\exists l \in L : l \in N_u\Big) \tag{126}$$

Then there exist two vertices u_1 and u_2 in U and two vertices l_1 and l_2 in L such that u_1 is adjacent to l_1 but not to l_2 and u_2 is adjacent to l_2 but not to l_1. In other words the induced subgraph is of the following form

$$G[\{u_1, u_2, l_1, l_2\}] = \tag{127}$$



where the dashed edges are individually either present or not. ◇

Proof. We will first show that $|L| \geq 2$ and $|U| \geq 2$. Pick an element $u_1 \in U$, which exists since $U \neq \emptyset$, by assumption there is a $l_1 \in L$ which is adjacent to u_1. Furthermore there exist a $u_2 \in U$ which is not adjacent to l_1, thus $u_1 \neq u_2$. Finally, by assumption there is a $l_2 \in L$ which is adjacent to u_2, thus $l_1 \neq l_2$. Note, that this does not yet prove the theorem, since u_1 and l_2 might be adjacent.

We will first prove the theorem for $|L| = 2$ and then use this to prove the general case.

$|L| = 2$:

Let's denote the vertices in L by l_1 and l_2. We first show by contradiction that both vertices in L must have at least one neighbor in U. Assume that l_1 does not have a neighbor in U, then all vertices in U must be adjacent to l_2 by the second part of eq. (126), but then the first part of eq. (126) is false. Thus l_1 has at least one neighbor in U and by symmetry the same is true for l_2. Now choose such a neighbor of l_1 in U and denote this u_1. We now show by contradiction that there exist another vertex $u_2 \in U$ which is adjacent to l_2 but not to l_1. Assume that this is not the case, i.e. all vertices in $U \setminus \{u_1\}$ are adjacent to l_1 or not adjacent to l_2. If a vertex in U is not adjacent to l_2 then it is necessarily adjacent to l_1, thus by assumption all vertices in $U \setminus \{u_1\}$ are adjacent to l_1. This is in contradiction with the first part of eq. (126) and the theorem for $|L| = 2$ follows.

$|L| > 2$:

We will show that the following is true: (1) G has an induced subgraph as in eq. (127) or (2) there exist a $l \in L$ such that $G \setminus l$ satisfy $\mathcal{R}(U, L \setminus \{l\})$. The theorem then follows since if (1) is true the theorem follows directly and if (2) is true we can make the same argument for $G \setminus l$ for some $l \in L$ and then possibly for $(G \setminus l) \setminus l'$ etc., which at some point will give the case $|L| = 2$, which we have proven above. Note that if the graph reached by deleting vertices from G, has the graph in eq. (127) as an induced subgraph, then so does G.

To prove that (1) or (2) is true, we show that if (2) is false then (1) is necessarily true. Therefore, assume now that (2) is false, which means that for every choice of l, $G \setminus l$ does not satisfy $\mathcal{R}(U, L \setminus \{l\})$. The only possibility for this to happen, i.e. the deletion of l makes the graph not satisfy eq. (126), is if the deletion of l makes some $u \in U$ not adjacent to any vertex in L. It is easy to see that this can only happen if $\exists u \in U : (L \cap N_u = \{l\})$. Since this should be true for all $l \in L$, we have that if (2) is false, the following is true,

$$\forall l \in L : \Big(\exists u \in U : (L \cap N_u = \{l\})\Big). \tag{128}$$

But eq. (128) implies that (1) is true. To see this pick two different vertices l_1 and l_2 in L. From eq. (128) we know that there exist a vertex $u_1 \in U$ such that $L \cap N_{u_1} = \{l_1\}$ and similarly a u_2 for l_2. Note that $u_1 \neq u_2$ since $N_{u_1} \neq N_{u_2}$. Furthermore, since $L \cap N_{u_1} = \{l_1\}$ and $L \cap N_{u_2} = \{l_2\}$ the induced subgraph $G[\{u_1, u_2, l_1, l_2\}]$ is as in eq. (127).

□

4.1.3.5 Proof for a complete-star graph

Here we prove that if $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \not\leq G$ for the case where $G[V']$ is a complete-star graph. We have the following theorem.

Theorem 4.5. *Let's assume that G is a distance-hereditary graph and V' is a subset $V' \subseteq V(G)$ such that the induced subgraph $G[V']$ is a complete-star graph $KS_{(B',L',c')}$. Furthermore assume that $\mathcal{P}(B', L', c')$ is false, then $S_{V'} \not< G$.* $\diamond$

Proof. We will prove this by induction on the size of B'. The base-case, $|B'| = 2$, is true due to theorem 4.3, since for $|B'| = 2$, the graph G is also a star-star graph. Let's now assume that theorem 4.5 is true for $|B'| = k$. We will now prove that the theorem is true for $|B'| = k + 1$ by showing that

$$\neg\mathcal{P}(B', L', c') \quad \Rightarrow \quad \exists b \in B' : \neg\mathcal{P}(B' \setminus \{b\}, L', c'). \tag{129}$$

where $|B'| = k + 1 \geq 3$. Then from the induction hypothesis we know that $S_{V'\setminus\{b\}} \not< G$ for some $b \in B'$ and the corollary follows, because if $S_{V'\setminus\{b\}}$ is not a vertex-minor of G then neither is $S_{V'}$, since $S_{V'\setminus\{b\}} < S_{V'}$. To show that eq. (129) is true, we instead show the contrapositive statement, i.e.

$$\forall b \in B' : \mathcal{P}(B' \setminus \{b\}, L', c') \quad \Rightarrow \quad \mathcal{P}(B', L', c'). \tag{130}$$

Let's therefore assume that $\mathcal{P}(B'\setminus\{b\}, L', c')$ is true for all b in B'. Let $\mathcal{Q}(u, B', L', c')$ be the expression on $\mathcal{P}(B', L', c')$ such that

$$\mathcal{P}(B', L', c') = \exists u \in V \setminus V' : \mathcal{Q}(u, B', L', c') \tag{131}$$

For each $b \in B'$, let u_b be a vertex in $V \setminus (V' \setminus \{b\})$ such that $\mathcal{Q}(u_b, B' \setminus \{b\}, L', c')$ is true. We now need to show that for at least one of these u_b, $u_b \in V \setminus V'$ and $\mathcal{Q}(u_b, B', L', c')$ is true. Note that $L' \cap N_{u_b} = \emptyset$ for all b, since $\mathcal{Q}(u_b, B' \setminus \{b\}, L', c')$ for all b. Thus, to show that for at least on u_b, $u_b \in V \setminus V'$ and $\mathcal{Q}(u_b, B', L', c')$ is true we will go trough the following steps:

1. Show that there can maximally be one $\tilde{b} \in B'$ such that $u_{\tilde{b}} \in B'$, i.e.

$$\exists\tilde{b} \in B' : \Big(\forall b \in B' : \big(b = \tilde{b} \ \vee \ u_b \notin B'\big)\Big) \tag{132}$$

2. Fix $\tilde{b} \in B'$ to be such that the corresponding expression in eq. (132) is true.[17]
3. Show that there can maximally be one $\hat{b} \in B' \setminus \{\tilde{b}\}$ such that $B' \nsubseteq N_{u_{\hat{b}}}$, i.e.

$$\exists\hat{b} \in B' \setminus \{\tilde{b}\} : \Big(\forall b \in B' \setminus \{\tilde{b}\} : \big(b = \hat{b} \ \vee \ B' \subseteq N_{u_b}\big)\Big) \tag{133}$$

4. Fix $\hat{b} \in B'$ to be such that the corresponding expression in eq. (133) is true.
5. Use step 1 to 4 to show that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and

$$(u_b, c') \notin E(G) \ \vee \ \exists h_b : \left(h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V'\setminus\{c'\}} N_x\right) \tag{134}$$

   i.e. $\mathcal{Q}(u_b, B', L', c')$ is true.

Let us now consider the steps 1 through 5 one by one.[18] We will in these steps often claim that certain small graphs are not distance-hereditary. Verifying this can be done by hand or using our code supplied at [1].

Step 1:

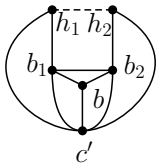
Here we show that eq. (132) is true. Firstly, if for all $b \in B'$ we have that $u_b \notin B'$, then eq. (132) is clearly true, since $\tilde{b}$ can then be chosen as any element in B'.[19] We now show by contradiction that there cannot exist two different vertices $\tilde{b}_1, \tilde{b}_2 \in B'$, such that $u_{\tilde{b}_1} \in B'$ and $u_{\tilde{b}_2} \in B'$. Thus, let's assume that such vertices $\tilde{b}_i$ for $i \in \{1, 2\}$, does exist. Note that, since $\mathcal{Q}(u_{\tilde{b}_i}, B' \setminus \{\tilde{b}_i\}, L', c')$ is true, we know that $u_{\tilde{b}_i}$ is adjacent to all vertices in $B' \setminus \{\tilde{b}_i\}$ and therefore $u_{\tilde{b}_i} = \tilde{b}_i$. Since $u_{\tilde{b}_i} = \tilde{b}_i$, we know that $u_{\tilde{b}_1} \neq u_{\tilde{b}_1}$. Furthermore, from the fact that $u_{\tilde{b}_i} \in B'$, we know that $u_{\tilde{b}_i}$ is adjacent to c' and thus, since $\mathcal{Q}(u_{\tilde{b}_i}, B' \setminus \{\tilde{b}_i\}, L', c')$ is true, there exist a vertex h_i such that

$$h_i \in N_{u_{\tilde{b}_i}} \cap N_{c'} \setminus \bigcup_{x \in V'\setminus\{c', \tilde{b}_i\}} N_x. \tag{135}$$

[17] Note that this does not imply that $u_{\tilde{b}} \in B'$.
[18] Step 2 and 4 are trivial.
[19] Remember that $|B'| \geq 3$.

The vertices h_i are necessarily different, since h_1 is adjacent to $b_1 = u_{\tilde{b}_1}$ but not to b_2 and vice versa for h_2. Now consider the following induced subgraph

$$G[\{c', \tilde{b}_1, \tilde{b}_2, b, h_1, h_2\}] = \tag{136}$$



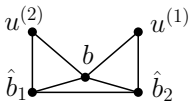
where b is a vertex in $B' \setminus \{b_1, b_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (136) is not distance-hereditary, independently if the edge (h_1, h_2) is present or not. Since this contradicts the fact that G is distance-hereditary, we know that eq. (132) is true. ◇

Step 3:

Here we show that eq. (133) is true. Firstly, if for all $b \in B' \setminus \{\tilde{b}\}$ we have that $B' \subseteq N_{u_b}$, then eq. (133) is clearly true, since $\hat{b}$ can then be chosen as any element in $B' \setminus \{\tilde{b}\}$.[20] We now show by contradiction that there cannot exist two different vertices $\hat{b}_1, \hat{b}_2 \in B'$, such that $B' \nsubseteq N u_{\hat{b}_1}$ and $B' \nsubseteq N u_{\hat{b}_2}$. Thus, let's assume that such vertices $\hat{b}_i$ for $i \in \{1, 2\}$, does exist. Let's for the remainder of this step denote $u_{\hat{b}_i}$ as $u^{(i)}$. From the previous steps we know that $u^{(i)} \notin B'$ and furthermore $(B' \setminus \{\hat{b}_i\}) \subseteq N_{u^{(i)}}$ since $\mathcal{Q}(u^{(i)}, B' \setminus \{\hat{b}_i\}, L', c')$ is true. Thus, by assumption, we have that $\hat{b}_i \notin N_{u^{(i)}}$. The vertices $u^{(i)}$ are then necessarily different, i.e. $u^{(1)} \neq u^{(2)}$, since for example $\hat{b}_1$ is a neighbor of $u^{(2)}$ but not of $u^{(1)}$. We will now show that this contradicts the fact that G is distance-hereditary, by considering the following cases:

- Assume that $u^{(1)}$ is not adjacent to $u^{(2)}$ and consider the following induced subgraph

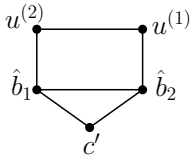
$$G[\{b, \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}\}] = \tag{137}$$



  where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (137) is not distance-hereditary.

- Assume that $u^{(1)}$ is adjacent to $u^{(2)}$.
  - Assume that neither $u^{(1)}$ or $u^{(2)}$ is adjacent to c' and consider the following induced subgraph

$$G[\{c', \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}\}] = \tag{138}$$



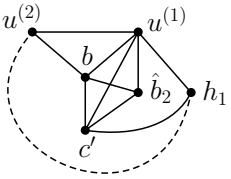
    which is not distance-hereditary.

  - Assume that exactly one of $u^{(1)}$ and $u^{(2)}$ is adjacent to c' and let's assume without loss of generality that it is $u^{(1)}$ that is adjacent to c'. Since $\mathcal{Q}(u^{(1)}, B' \setminus \{\hat{b}_1\}, L', c')$ is true and $u^{(1)}$ is adjacent to c', we know that there exist a vertex h_1 such that

$$h_1 \in N_{u^{(1)}} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', b_1\}} N_x. \tag{139}$$

    Consider now the following induced subgraph

$$G[\{c', b, \hat{b}_2, u^{(1)}, u^{(2)}, h_1\}] = \tag{140}$$



    where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (140) is not distance-hereditary, independently if the edge $(h_1, u^{(2)})$ is present or not.

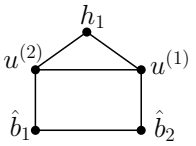
[20] Remember that $|B'| \geq 3$.

– Assume that both $u^{(1)}$ and $u^{(2)}$ is adjacent to c'. Since $\mathcal{Q}(u^{(i)}, B' \setminus \{\hat{b}_i\}, L', c')$ is true and $u^{(i)}$ is adjacent to c', we know that there exist a vertex h_i such that

$$h_i \in N_{u^{(i)}} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', b_i\}} N_x. \tag{141}$$

* Assume that $h_1 = h_2$, which implies that h_1 is not adjacent to $\hat{b}_1$ or $\hat{b}_2$ and consider the following induced subgraph

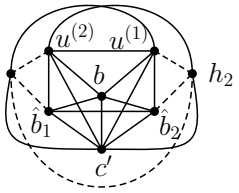
$$G[\{\hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}, h_1\}] = \tag{142}$$



which is not distance-hereditary.

* Assume that $h_1 \neq h_2$ and consider the following induced subgraph

$$G[\{c', b, \hat{b}_1, \hat{b}_2, u^{(1)}, u^{(2)}, h_1, h_2\}] = \tag{143}$$



where b is a vertex in $B' \setminus \{\hat{b}_1, \hat{b}_2\}$, which exists since $|B'| \geq 3$ by assumption. The graph in eq. (143) is not distance-hereditary, independently if the edges

$$(h_1, h_2),\ (h_1, u^{(2)}),\ (h_2, \hat{b}_2),\ (h_1, u^{(1)}),\ (h_1, \hat{b}_2) \tag{144}$$

are individually present or not. To make this statement more transparent, we also provide the adjacency matrix of the graph in eq. (143). The graph in eq. (143) has adjacency matrix

$$\Gamma = \begin{array}{c} \\ c' \\ b \\ \hat{b}_1 \\ \hat{b}_2 \\ u^{(1)} \\ u^{(2)} \\ h_1 \\ h_2 \end{array} \begin{array}{c} \begin{array}{cccccccc} c' & b & \hat{b}_1 & \hat{b}_2 & u^{(1)} & u^{(2)} & h_1 & h_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & x_1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & x_2 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & x_3 \\ 1 & 1 & 1 & 0 & 1 & 0 & x_4 & 1 \\ 1 & 0 & x_1 & 0 & 1 & x_4 & 0 & x_5 \\ 1 & 0 & 0 & x_2 & x_3 & 1 & x_5 & 0 \end{pmatrix} \end{array} \tag{145}$$

where $x_1, \ldots, x_5 \in \{0, 1\}$. By explicit computation one can check that for any assignment of the variables $x_1, \ldots, x_5$, the graph with adjacency matrix as in eq. (145) is not distance-hereditary.

Since in all cases we arrived at a contradiction of the fact that G is distance-hereditary, we know that eq. (133) is true. ◇

Step 5:

Here we show that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and

$$(u_b, c') \notin E(G) \ \vee \ \exists h_b : \left(h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right). \tag{146}$$

Note that $B' \subseteq N_{u_b}$ implies $u_b \notin B'$, thus we can focus on the first property. We prove the statement by contradiction and assume therefore that there exist no such b, i.e. there exist no $b \in B'$ such that $B' \subseteq N_{u_b}$ and for which eq. (146) is true. Let's first introduce the set $\mathcal{B}$ of vertices in B' which satisfy the first of these properties, i.e.

$$\mathcal{B} = \{b \in B' : (B' \subseteq N_{u_b})\}. \tag{147}$$

From the previous steps we know that $\mathcal{B}$ is not empty. Furthermore, from our assumption we must have that for all $b \in \mathcal{B}$, eq. (146) is false, i.e

$$\forall b \in \mathcal{B} : \left((u_b, c') \in E(G) \ \wedge \ \forall h_b : \left(h_b \notin N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c'\}} N_x \right) \right). \tag{148}$$

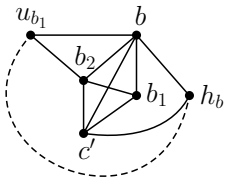
Let b now be a fixed element of $\mathcal{B}$. Since u_b is adjacent to c' and from the fact that $\mathcal{Q}(u_b, B' \setminus \{b\}, L', c')$ is true, we know that there exist a h_b such that

$$h_b \in N_{u_b} \cap N_{c'} \setminus \bigcup_{x \in V' \setminus \{c', b\}} N_x. \tag{149}$$

Note that, eq. (148) together with eq. (149) implies that h_b is adjacent to b but to no other vertex in B'. We will now show that this leads to a contradiction by considering a vertex $b_1 \in B' \setminus \{b\}$, such that $u_{b_1} \notin B'$, which we showed exists in step 1. Furthermore, let b_2 a vertex in $B' \setminus \{b, b_1\}$, which is necessarily adjacent to u_{b_1}, since $\mathcal{Q}(u_{b_1}, B' \setminus \{b_1\}, L', c')$ is true. Let's consider the following cases:

- Assume that u_{b_1} is not adjacent to c'. By assumption we then have that $B' \nsubseteq N_{u_{b_1}}$, which implies that u_{b_1} is not adjacent to b_1. Consider now the following induced subgraph

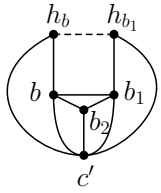
$$G[\{c', b, b_1, b_2, u_{b_1}, h_b\}] = \tag{150}$$



  which is not distance-hereditary, independently if the edge (h_b, u_{b_1}) is present or not.

- Assume that u_{b_1} is adjacent to c'.
  - Assume that $B' \subseteq N_{u_{b_1}}$. By the same argument as for b and u_b, we know that there exist a vertex h_{b_1} which is adjacent to u_{b_1}, c' and b_1 but not to any other vertex in B'. Consider now the following induced subgraph

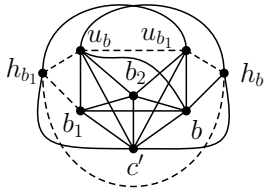
$$G[\{c', b, b_1, b_2, h_b, h_{b_1}\}] = \tag{151}$$



    which is not distance-hereditary, independently if the edge (h_b, h_{b_1}) is present or not.

  - Assume that $B \nsubseteq N_{u_{b_1}}$, which implies that b_1 is not adjacent to u_{b_1} since $\mathcal{Q}(u_{b_1}, B' \setminus \{b_1\}, L', c')$ is true. Furthermore, we know that there is a vertex h_{b_1} which is adjacent to u_{b_1} and c' and possibly to b_1 but no other vertex in B'. Consider now the following induced subgraph

$$G[\{c', b, b_1, b_2, u_b, u_{b_1}, h_b, h_{b_1}\}] = \tag{152}$$



    which is not distance-hereditary, independently if the edges

$$(h_b, h_{b_1}),\ (h_b, u_{b_1}),\ (h_{b_1}, b_1),\ (h_{b_1}, u_b),\ (u_b, u_{b_1}) \tag{153}$$

    are individually present or not. To make this statement more transparent, we also provide the adjacency

matrix of the graph in eq. (152). The graph in eq. (152) has adjacency matrix

$$\Gamma = \begin{array}{c} \\ c' \\ b \\ b_1 \\ b_2 \\ u_b \\ u_{b_1} \\ h_b \\ h_{b_1} \end{array} \begin{array}{c} \begin{array}{cccccccc} c' & b & b_1 & b_2 & u_b & u_{b_1} & h_b & h_{b_1} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & x_1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & x_2 & 1 & x_3 \\ 1 & 1 & 0 & 1 & x_2 & 0 & x_4 & 1 \\ 1 & 1 & 0 & 0 & 1 & x_4 & 0 & x_5 \\ 1 & 0 & x_1 & 0 & x_3 & 1 & x_5 & 0 \end{pmatrix} \end{array} \tag{154}$$

where $x_1, \ldots, x_5 \in \mathbb{F}_2$. By explicit computation one can check that for any assignment of the variables $x_1, \ldots, x_5$, the graph with adjacency matrix as in eq. (154) is not distance-hereditary.

Since in all cases we arrived at a contradiction of the fact that G is distance-hereditary, we know that there exist a $b \in B'$ such that $u_b \notin B'$, $B' \subseteq N_{u_b}$ and such that eq. (146) is true. ◇

We have therefore shown that eq. (130) is true which implies that eq. (129) is true. Finally, as we described in the beginning of the proof, this implies, by induction, that if $\mathcal{P}(B', L', c')$ is false then $S_{V'} \not\leq G$. □

4.2 Fixed-parameter tractable algorithm for unbounded rank-width

In this section we will show that the star vertex-minor problem (STARVERTEXMINOR) is fixed-parameter tractable for circle graphs, in terms of the size of the considered star graph. More specifically, we will show that there exists an efficient algorithm to decide if $S_{V'}$ is a vertex-minor of G, given that G is a circle graph and the subset $V' \subseteq V(G)$ is of size k. We will do this by showing that we can map this problem in polynomial time to deciding whether the 4-regular multi-graph that defines G has a SOET on the vertices V'. This is done in section 4.2.1. We will then give an algorithm that decides whether a 4-regular multi-graph has a SOET on a subset V' of its vertices where $V' = k$ is fixed. This is done in section 4.2.2. We begin by formally stating the decisions problems considered.

We first define the problem $k-$STARVERTEXMINOR.

Problem 4.1 ($k-$STARVERTEXMINOR)**.** *Let G be a graph and let V' be a subset of $V(G)$ with $|V'| = k$. Decide whether $S_{V'}$ is a vertex-minor of G.* ◇

We also define the problem $k-$SOET.

Problem 4.2 ($k-$SOET)**.** *Let F be a 4-regular multi-graph and let V' be a subset of $V(F)$ with $|V'| = k$. Decide whether F allows for a* SOET *with respect to V'* ◇

Theorem 4.6. $k-$STARVERTEXMINOR*, restricted to circle graphs, is in $\mathbb{P}$.* ◇

Proof. This will follow from theorem 4.7 which provides an efficient mapping of every circle graph instance of $k-$STARVERTEXMINOR to a corresponding instance of $k-$SOET. By corollary 4.9.1 $k-$SOET is in $\mathbb{P}$ and hence so is $k-$STARVERTEXMINOR. An efficient algorithm for $k-$SOET is given in algorithm 4. □

This theorem has the following corollary

Corollary 4.6.1. STARVERTEXMINOR *is fixed-parameter tractable in the size of the input vertex-set V' if the input graph G is a circle graph.* ◇

The existence of this fixed-parameter tractable algorithm is theoretically interesting but it is not likely to be of practical use. This is so because while the algorithm is efficient in the size of the input graph G it suffers from a hidden constant that is of size $O(k! \cdot (f(k)^{\mathcal{O}(kf(k))}))$, where $f(k) = 2^{2^{\mathcal{O}(k^2)}}$ making its practical implementation unlikely.

4.2.1 Mapping $k-$STARVERTEXMINOR to $k-$SOET

In this section we will prove that there exists an efficient mapping from instances of $k-$STARVERTEXMINOR that are circle graphs to $k-$SOET. This is formalized in the following theorem, the proof of which also provides a prescription of the algorithm that defines the mapping.

Theorem 4.7. *Let (G, V') be an instance of $k-$*STARVERTEXMINOR *and let G be a circle graph. These is an efficient mapping from this instance to an instance of $k-$*SOET *and moreover the instance (G, V') is a* yes*-instance of $k-$*STARVERTEXMINOR *if and only if its image under the mapping is a* yes*-instance of $k-$*SOET*.* ◇

Proof. We will prove this by providing an explicit mapping. An instance (G, V') of $k-$STARVERTEXMINOR, where G is a circle graph and V' a vertex set, can be mapped to an instance of $k-$SOET by the following two steps:

- Find a double occurrence word $\boldsymbol{X}$ with letters in $V(G)$ such that $G = \mathcal{A}(\boldsymbol{X})$. This can be done in time $\mathcal{O}(|V(G)|^2)$ by using Spinrad's algorithm [46].
- Construct a 4-regular multi-graph F, such that $\boldsymbol{X} = m(U)$ for some Eulerian tour U on F. As shown in [12], this can be done in the following way:
  - Let $C_{\boldsymbol{X}}$ be a cycle graph with the vertices labeled as the consecutive letters of $\boldsymbol{X}$.
  - Contract every pair of vertices with the same label, while keeping all the edges. Note that this step can create multi-edges or self-loops. This step can be done in time $\mathcal{O}(|V(G)|)$ by adding the corresponding rows of the adjacency matrix and deleting one row and one column.

  The graph obtained from these steps is then a 4-regular multi-graph F with a Eulerian tour U, such that $\boldsymbol{X} = m(U)$. Therefore, constructing the 4-regular multi-graph F, given $\boldsymbol{X}$, can also be done in time $\mathcal{O}(|V(G)|^2)$.

One can see that the above algorithm runs in $\mathcal{O}(|V(G)|^2)$ and given a circle graph G outputs a 4-regular graph F that has a Eulerian tour U such that $\mathcal{A}(U) = G$. From corollary 2.6.1 we then know that $S_{V'} < G$ if and only if F allows for a SOET with respect to V'. Using the above two steps we see that any circle graph instance of $k-$STARVERTEXMINOR can be mapped to $k-$SOET in time $\mathcal{O}(|V(G)|^2)$. □

Given this mapping, the next logical step is to find an efficient algorithm for $k-$SOET. This is done in the next section.

4.2.2 $k-$SOET is in $\mathbb{P}$

In this section we will prove that $k-$SOET is in $\mathbb{P}$. We will do this by explicitly writing down an efficient algorithm. This algorithm will make use of an algorithm which solves a well known graph problem we call $k'-$DPP. $k'-$DPP, for k'-Disjoint Path Problem is formally defined as follows.

Problem 4.3 ($k'-$DPP)**.** *Let H be a multi-graph and let $L = \{(v_1, v_1'), \ldots, (v_{k'}, v_{k'}')\}$ be a set of two-tuples of vertices of H such that $|L| = k'$. Decide whether there exist k' edge-disjoint paths P_i on H that start at v_i and end at v_i' for $i \in [k']$.* ◇

Robertson and Seymour proved that there exist an efficient algorithm for solving $k'-$DPP, if k' is fixed. Their proof of correctness is based on 23 papers named Graph minors. I , ..., Graph minors. XXIII [44]. In [31] an improved algorithm for $k'-$DPP is given, with a much smaller hidden constant describing the scaling in terms of k'. This hidden constant is still quite large as running time for the algorithm in [31] is $(f(k')^{\mathcal{O}(k' f(k'))})n^{\mathcal{O}(1)}$, where n is the number of vertices in the graph and $f(k') = 2^{2^{\mathcal{O}(k'^2)}}$

We will discuss an algorithm that solves $k-$SOET efficiently. It will use an algorithm for $k'-$DPP as a subroutine, calling it a constant number of times.

The algorithm for $k-$SOET is sketched as follows. Let F be a 4-regular multi-graph and let $V' \subset V(F)$ be of size k. Recall that a SOET is a Eulerian path that visits the vertices in V' in some order. A necessary condition for a SOET to exist is that there are, for some ordering of the vertices in V' two edge-disjoint trails from the first vertex in V' to the second vertex in V' and from the second to the third and so on. If one is given such a collection of paths it

is not hard to see that one can connect these paths to each other to form a tour and moreover extend this total tour to be Eulerian. Hence we could use the $k'-$DPP algorithm described above to to find such a tour by finding all the edge disjoint trails that connect the vertices in V'.

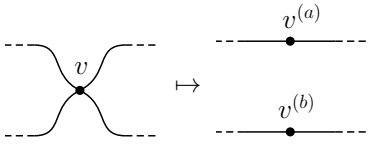
There are two problems with this. The first problem is that a SOET with respect to V' can exist with respect to any possible ordering of the set V'. This means that in order to find this tour we might have to apply $k-$DPP to all $k!$ different orderings. This is a large overhead but acceptable since we are only looking for an algorithm that is efficient for fixed k. The second problem is that the $k'-$DPP algorithm expects k' pairs of vertices and requires these pairs to be different.

This means we cannot input the same vertex-pair twice to get two edge disjoint paths. We can resolve this at some overhead by running the $k'-$DPP algorithm on a modified multi-graph H. This multi-graph, which we call a SOET splitting of F, is created by taking each vertex v_i in V and splitting it into two vertices $v_i^{(a)}, v_i^{(b)}$ such that the four edges $e_1, \dots e_4$ originally incident on v_i are now pairwise incident on $v_i^{(a)}, v_i^{(b)}$.

As an example we could have for instance that e_1, e_2 are incident on $v_i^{(a)}$ and e_3, e_4 are incident on $v_i^{(b)}$. We must consider all possible choices of pairings here (of which there are 6) and since there are k vertices on which to perform this procedure (each vertex in V') there are 6^k such SOET splittings of F (which are not 4-regular anymore). This means we must call the $2k-$DPP subroutine $6^k \cdot k!$ times to account for all possible orderings of the SOET.

Let's make this a little more rigorous. We begin by defining the notion of a SOET-splitting of a 4-regular multi-graph F with respect to a set V'.

Definition 4.3 (SOET-splitting)**.** *Let F be a 4-regular multi-graph and let V' be a subset of it's vertices. A* SOET*-splitting H of F with respect to V is a multi-graph created from F y performing the following operation on all vertices in V*



(155)

We label the two vertices that originate from a vertex $v \in V'$ as $v^{(a)}$ and $v^{(b)}$. Note that we have not specified how to connect the edges that were originally incident on v to $v^{(a)}$ and $v^{(b)}$. There are six possible ways to do this for each $v \in V'$ and each choice leads to an a priori distinct SOET*-splitting multi-graph H.* ◇

We also define the subroutine $k'-$DPP

Algorithm 3 $k'-$DPP

1: **INPUT**: A multi-graph H and a set of 2-tuples $\{(v_1, \hat{v}_1), \dots (v_{k'}, \hat{v}_{k'})\}$
2: **OUTPUT**: TRUE if there exist edge-disjoint paths in H connecting $v_i, \hat{v}_i$ for all $i \in [k']$.
3: FALSE otherwise

Note that we have only specified the inputs and outputs of this subroutine. For details on the inner workings of this algorithm see [31]. Using this subroutine we can write down an algorithm for $k-$SOET.

Algorithm 4 $k-$SOET

INPUT: A 4-regular multi-graph F and a set $V' = \{v_1, \ldots, v_k\}$ such that $V' \subset V(F)$
OUTPUT: TRUE if there exist a SOET on F with respect to V'
FALSE otherwise

Generate a list L of all possible SOET-splittings of F with respect to V'

for all graphs H in L **do**

  for all permutations π of the set $[1:k]$ **do**
    Run $2k-$DPP on the set $\{(v^{(a)}_{\pi(1)}, v^{(a)}_{\pi(2)}), \ldots, (v^{(a)}_{\pi(k)}, v^{(a)}_{\pi(1)}), (v^{(b)}_{\pi(1)}, v^{(b)}_{\pi(2)}), \ldots, (v^{(b)}_{\pi(k)}, v^{(b)}_{\pi(1)})\}$

    if $2k-$DPP returns TRUE **then**
      RETURN TRUE
      QUIT
    end if
  end for
end for
RETURN FALSE
QUIT

We now prove that this algorithm returns TRUE if and only if the multi-graph F allows for a SOET with respect to the vertex-subset V'. We begin by proving that if the algorithm returns TRUE the multi-graph F allows for a SOET with respect to the vertex-subset V'. We have the following theorem.

Theorem 4.8. *Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If algorithm 4 returns TRUE then F allows for a* SOET *with respect to the vertex set V'.* ◇

Proof. Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If algorithm 4 returns true this means there exists a SOET-splitting H of F and a permutation π of the set $[k]$ such that there exist $2k$ edge-disjoint paths in H that connect the vertices $v^{(a)}_{\pi(1)}$ and $v^{(a)}_{\pi(2)}$, $v^{(a)}_{\pi(2)}$ and $v^{(a)}_{\pi(3)}$ and so forth. Undoing the SOET-splitting operation that defines F we can see that these edge-disjoint paths can be attached to one another to form a closed trail[21] (a tour) U on F that visits all vertices of V' twice in the order $v_{\pi(1)}, \ldots, v_{\pi(k)}$. This is not yet a Eulerian tour however. To construct a Eulerian tour out of the tour U we consider the multi-graph $\hat{H}$ obtained from F by deleting all vertices in V (looking at the induced subgraph $F[V \setminus V']$) and subsequently removing all edges in the tour U from the remaining multi-graph. The resulting multi-graph will consist of multiple connected components. These connected components will either consist of a single vertex or will be multi-graphs with vertices of degree two and four. This means all these connected components are Eulerian. Moreover each of these connected components will contain a vertex that is also a vertex in the tour U. Consider on each of these connected components then a Eulerian tour. These tours will thus form tours on the original multi-graph F as well and since all such tours have at least one vertex in common with the tour U we can extend U to a Eulerian tour on the multi-graph F by for each connected component cutting U at such a vertex and inserting the tour originating from the connected components of G. This means that U can be turned into a Eulerian tour and hence there exists a SOET on the multi-graph F. This proves the theorem. □

Next we prove the converse statement.

Theorem 4.9. *Let F be a connected 4-regular multi-graph and let $V' \subset V(F)$ be a subset of its vertices with $|V'| = k$. If F allows for a* SOET *with respect to V' then algorithm 4 will return TRUE.* ◇

Proof. Let U be a SOET on F with respect to V' and without loss of generality assume that U traverses the vertices of V' in the order $v_1, v_2, \ldots, v_k$. Hence for the vertices v_1, v_2 there are 2 sub-trails $U^{(a)}_1, U^{(b)}_1$ of U that start at v_1 and end at v_2. The sub-trail $U^{(a)}_1$ (or $U^{(b)}_1$) might not be a path, but one can easily pick a subset of the edges in $U^{(a)}_1$

[21] Note that a path is also a trail.

which gives a path, for example as the shortest path $P_1^{(a)}$ between v_1 and v_2 in the subgraph of F given by the vertices and edges of $U_1^{(a)}$. Similarly for $U_1^{(b)}$ and $P_1^{(b)}$. We can make the same argument for the vertices v_2, v_3 and so forth. By the definition of SOET splittings there must thus exist a SOET splitting G of F with respect to V' such that the path $P_1^{(a)}$ starts at $v_1^{(a)}$ and ends at $v_2^{(a)}$ and such that the path $P_1^{(b)}$ starts at $v_1^{(b)}$ and ends at $v_2^{(b)}$. We can make the same argument for the vertices v_2, v_3 and so forth. Hence there must exist a SOET-splitting G of F with respect to V' such that the algorithm $2k-\text{DPP}(G, S)$ with $S = \{(v_1^{(a)}, v_2^{(a)}), \ldots, (v_k^{(a)}, v_1^{(a)}), (v_1^{(b)}, v_2^{(b)}), \ldots, (v_k^{(b)}, v_1^{(b)})\}$ returns TRUE. Since algorithm 4 runs over all possible orderings of V' and all possible SOET splittings this particular call to $2k-$DPP will always happen and hence algorithm 4 will return TRUE. This proves the theorem. □

This leads to the following corollary.

Corollary 4.9.1. *$k-$SOET is in $\mathbb{P}$.* ◇

Proof. By *theorem* 4.9 and *theorem* 4.8, *algorithm* 4 returns TRUE if and only if the tuple (F, V'), with F a 4-regular multi-graph and $V' \subset V(F)$ a subset of its vertices with $|V'| = k$, is a YES instance of $k-$SOET. Moreover the function $k'-$DPP runs in polynomial time in the size of the input graph and we have for all SOET-splittings G of F with respect to V' that $|V(G)| = |V(F)| + k$ and $|E(G)| = |E(F)|$. Algorithm 4 hence performs a constant number of function calls (constant in the size of F, not in k) of $k'-$DPP with an input multi-graph of size $O(|V(F)|, |E(F)|)$. Hence Algorithm 4 runs in polynomial time with respect to $|V(F)|, |E(F)|$ and thus we have that $k-$SOET is in $\mathbb{P}$. □

This corollary then leads to theorem 4.6.

5 Connected vertex-minor on three vertices or less

In this section we show that a fixed connected graph with vertices V' is a vertex-minor of a connected graph G if $V' \subseteq V(G)$ and $|V'| \leq 3$. Furthermore we provide an algorithm that reduces the number of operations one need to transform G to the desired graph.

Theorem 5.1. *Let G be a connected graph and G' be a connected graph with vertices V', such that $|V'| \leq 3$. Then we have that*

$$G' < G \quad \Leftrightarrow \quad V' \subseteq V(G) \tag{156}$$

◇

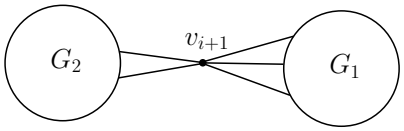
Proof. Note that if $G' < G$ then clearly $V' \subseteq V(G)$. Assume therefore that $V' \subseteq V(G)$. Let's denote the vertices in $V(G) \backslash V'$ as $U = (v_1, v_2, \ldots, v_{n-k}$, where $n = |V(G)|$ and $k = |G'|$. We will now show that $G' < G$ by finding a sequence of operations $P = P_{v_{n-k}} \circ \cdots \circ P_{v_1}$, where $P_{v_i} \in \{X_v, Y_v, Z_v\}$, such that each intermediate graph $G^{(i)} = P_{v_i} \circ \cdots \circ P_{v_1}(G)$ is connected. Since any connected graph with vertices V', for $|V'| \leq 3$, is either a star graph or a complete graph, which are LC-equivalent, this shows that $G' < G$. Let's therefore consider such an intermediate graph $G^{(i)}$ for some $i \in [n-k]$ and the next vertex v_{i+1}. We will now show that $G^{(i)} \setminus v_{i+1}$ or $\tau_{v_{i+1}}(G^{(i)}) \setminus v_{i+1}$ is connected. This will be done by considering the following two cases:

Assume that $G^{(i)} \setminus v_{i+1}$ is not connected:

This means that v_{i+1} is a cut vertex. Let G_1 be a connected component in the graph $G^{(i)} \setminus v_{i+1}$ and G_2 the rest of the vertices, see eqs. (157) and (158) for an illustration. Since v_{i+1} is a cut vertex, we know that no vertex in $N_{v_{i+1}} \cap V(G_2)$ is adjacent to any vertex in $N_{v_{i+1}} \cap V(G_1)$, in the graph $G^{(i)}$. Thus, all vertices in $N_{v_{i+1}} \cap V(G_2)$ are adjacent to all vertices in $N_{v_{i+1}} \cap V(G_1)$ in the graph $\tau_{v_{i+1}}(\hat{G}) \setminus v_{i+1}$, showing that this graph is connected, see

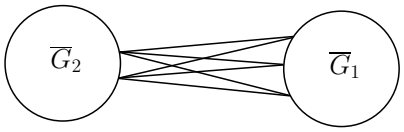
eq. (159).

$$G^{(i)} = \quad (157)$$



$$G^{(i)} \setminus v_{i+1} = \quad (158)$$

$$\tau_{v_{i+1}}(G^{(i)}) \setminus v_{i+1} = \quad (159)$$



Assume that $\tau_v(\hat{G}) \setminus v$ is not connected:

This case follows from the previous. To see this, let $\tilde{G}$ be the graph $\tau_v(G^{(i)})$. From the above case we then know that $\tau_v(\tilde{G}) \setminus v_{i+1}$ is connected. But this graph is exactly $G^{(i)} \setminus v_{i+1}$ since local complementations are involutions and the theorem follows.

□

From theorem 5.1 we can easily formulate an algorithm that finds a sequence of operations taking G to a desired connected graph G' on V', by simply checking recursively if $\hat{G} \setminus v$ or $\tau_v(\hat{G}) \setminus v$ is connected. However, if the vertices in V' are already close in G and G is a very large, it would be practical to not have to consider all the vertices in G. Next, we present a more practical algorithm to find a sequence of local complementations and vertex-deletions that take some graph G to a star graph[22] on the vertices a, b and c. The algorithm performs the following steps:

- Find the shortest path P_1 between a and b and connect these by doing τ-operations along this path. This first step already give an algorithm for creating a star graph on vertices a and b, see algorithm 5.
- Then do the same with b and c by finding the shortest path P_2 between b and c. The question is now whether we removed the edge between a and b while connecting b and c. This could only happen if P_2 goes through a vertex which is a common neighbor of a and b. Furthermore, since P_2 is the shortest path from c to b this could only be the case for the last vertex on the path, before b.
  - If c is connected to a before the last local complementation on P_2, then stop, since the induced graph is already connected.
  - Assume that c is not connected to a before the last local complementation is performed. So in this case, after performing the local complementation along P_2, a and b will not be connected but they will both be connected to c.

The full protocol is formalized in algorithm 6.

Algorithm 5 Producing S_1 on vertices a and b

1: **INPUT**: A graph G and two vertices $a, b \in V(G)$.
2: **OUTPUT**: A sequence of vertices $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[\{a, b\}]$ is a star graph.
3:
4:
5: Find the shortest path P between a and b
6: Perform τ_p for all $p \in P \setminus \{a, b\}$

[22] Note that any other connected graph on $\{a, b, c\}$ can easily be constructed.

Algorithm 6 Producing S_2 on vertices a, b and c

INPUT: A graph G and three vertices $a, b, c \in V(G)$.
OUTPUT: A sequence of vertices $\boldsymbol{v}$ such that $\tau_{\boldsymbol{v}}(G)[\{a, b, c\}]$ is a star graph.

Find the shortest path P_1 between a and b
Perform τ_p for all $p \in P_1 \setminus \{a, b\}$
Find the shortest path $P_2 = (p_0 = b, p_1, \ldots, p_n, p_{n+1} = c)$
for i **in** $1, \ldots, n$ **do**
    if a and c are not adjacent **then**
        Perform τ_{p_i}
    end if
end for

6 Conclusion

We have shown that deciding if a graph state $|G'\rangle$ can be obtained from another graph state $|G\rangle$ using LC+LPM+CC is $\mathbb{NP}$-Complete, by showing that VERTEXMINOR is $\mathbb{NP}$-Complete. The computational complexity of VERTEXMINOR was previously unknown and was posted as an open problem in [18]. It is important to note that our results are for labeled graphs, since vertices correspond to physical qubits in the case of transforming graph states.

We presented an efficient algorithm for STARVERTEXMINOR when the input graph is restricted to be distance-hereditary. It would be interesting to know if the same approach generalize to qudit graph states described by weighted graphs of low rank-width.

We presented an efficient algorithm for $k-$STARVERTEXMINOR on circle graphs, that is the problem of deciding if the star graph on a subset of vertices V', with $|V'| = k$, is a vertex-minor of another graph. However, the computational complexity of $k-$STARVERTEXMINOR or $k-$VERTEXMINOR on general graphs is still unknown.

Below we list some more open questions regarding the computational complexity of deciding how graph states can be transformed using local operations:

- Is the problem still $\mathbb{NP}$-Complete if one allows for arbitrary local operations and classical communication (LOCC), instead of restricting to only LC + LPM + CC? Is there an efficient algorithm for states with bounded Schmidt-rank width also in this case? It has been shown that many graph state are equivalent under LC if and only if they are equivalent under stochastic LOCC (SLOCC) [51]. An interesting question is therefore whether graph states described by distance-hereditary or circle graphs fall into this category.
- What is the computational complexity of deciding if a graph state $|G\rangle$ can be transformed in to another $|G'\rangle$ if the qubits are partitioned into multiple local sets, within which multi-qubit Clifford operations and multi-qubit Pauli measurements can be performed.
- How do the computational complexity results of this paper relate to the complexity of *code switching* in stabilizer quantum error correction codes, e.g. fault-tolerant interconversion between two stabilizer codes [27]? More precisely, can one re-use the techniques presented here, to prove that it is computationally hard to decide whether one stabilizer code can be fault-tolerantly transformed into another. We conjecture that this could be possible, since many stabilizer codes (such as topological stabilizer codes) have a notion of 'locality' with fault-tolerant conversions being the conversions that respect this locality [37].

While finishing this work we became aware of work by F. Hahn et al [29], where specific instances of VERTEXMINOR are considered in the context of quantum network routing.

Acknowledgements

We thank Kenneth Goodenough, Think Le, Bas Dirkse, Victoria Lipinska, Stefan Bäuml, Tim Coopmans, Jérémy Ribeiro, Ben Criger and Jens Eisert for discussions. AD, JH and SW were supported by STW Netherlands, NWO VIDI grant and an ERC Starting grant.

References

[1] Git-repository with implemented code. `https://github.com/AckslD/vertex-minors`.

[2] K. Azuma, K. Tamaki, and H.-K. Lo. All-photonic quantum repeaters. *Nature communications*, 6:6787, 2015.

[3] H.-J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41(2):182 – 208, 1986.

[4] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.*, 560(P1):7–11, 2014.

[5] U. Bertele and F. Brioschi. *Nonserial dynamic programming*. Academic Press, 1972.

[6] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1976.

[7] A. Bouchet. Caracterisation des symboles croises de genre nul. *Comptes Rendus de l'Académie des Sciences*, 274:724–727, 1972.

[8] A. Bouchet. Graphic presentations of isotropic systems. *Journal of Combinatorial Theory, Series B*, 45(1):58–76, 1988.

[9] A. Bouchet. Transforming trees by successive local complementations. *Journal of Graph Theory*, 12(2):195–207, 1988.

[10] A. Bouchet. *κ-Transformations, Local Complementations and Switching*, pages 41–50. Springer Netherlands, Dordrecht, 1990.

[11] A. Bouchet. An efficient algorithm to recognize locally equivalent graphs. *Combinatorica*, 11(4):315–329, 1991.

[12] A. Bouchet. Circle Graph Obstructions. *Journal of Combinatorial Theory, Series B*, 60(1):107–144, 1994.

[13] M. Christandl and S. Wehner. Quantum Anonymous Transmissions. In *Lecture Notes in Computer Science*, volume 3788 LNCS, pages 217–235. 2005.

[14] B. Courcelle. Circle graphs and monadic second-order logic. *Journal of Applied Logic*, 6(3):416–442, 2008.

[15] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language Theoretic Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[16] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of computer and system sciences*, 46(2):218–270, 1993.

[17] B. Courcelle and S. il Oum. Vertex-minors, monadic second-order logic, and a conjecture by Seese. *Journal of Combinatorial Theory. Series B*, 97(1):91–126, 2007.

[18] K. K. Dabrowski, F. Dross, J. Jeong, M. Kanté, O.-j. Kwon, S.-i. Oum, and D. Paulusma. Recognizing Small Pivot-Minors. In J. Q. Open and J. R. Access, editors, *42nd Conference on Very Important Topics (CVIT 2016)*, number 23, pages 23:1–23:15, 2016.

[19] A. Dahlberg and S. Wehner. Transforming graph states using single-qubit operations. *Phil. Trans. R. Soc. A 376, One contribution of 15 to a discussion meeting issue 'Foundations of quantum mechanics and their impact on contemporary society'* `http://dx.doi.org/10.1098/rsta.2017.0325,arxiv.org/abs/1805.05305`, 2018.

[20] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271, 1959.

[21] R. G. Downey and M. R. Fellows. Parameterized Complexity. *Proc 6th Annu Conf on Comput Learning Theory*, 5(1):51–57, 1999.

[22] A. K. Ekert. Quantum cryptography based on Bell's theorem. *Physical Review Letters*, 67(6):661, 1991.

[23] M. Fleury. Deux problemes de Geometrie de sitation. *Journal de mathematiques elementaires*, 2nd(2):257–261, 1883.

[24] R. Ganian and P. Hliněný. On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width. *Discrete Applied Mathematics*, 158(7):851–867, 2010.

[25] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[26] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. 2nd edition, 2004.

[27] D. Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, 1997.

[28] D. Gottesman. The Heisenberg Representation of Quantum Computers. *Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, 1:32–43, 1998.

[29] F. Hahn, A. Pappa, and J. Eisert. Quantum network routing and local complementation. *arXiv pre-print:* 1805.04559.

[30] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, H. J. Briegel, M. V. den Nest, and H. J. Briegel. Entanglement in Graph States and its Applications. *Quantum Computers, Algorithms and Chaos*, pages 1–99, 2006.

[31] K. I. Kawarabayashi and Y. Kobayashi. The edge-disjoint paths problem in Eulerian graphs and 4-edge-connected graphs. *Combinatorica*, 35(4):477–495, 2015.

[32] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin. A quantum network of clocks. *Nature Physics*, 10(8):582, 2014.

[33] A. Kotzig. Eulerian lines in finite 4-valent graphs and their transformations. In *Colloquium on Theory of Graphs (1966 : Tihany, Hungary)*, pages 219–230, 1968.

[34] A. Kotzig. Quelques remarques sur les transformations κ. In *seminaire Paris*, 1977.

[35] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Computer Science Review*, 13-14:39–74, 2014.

[36] D. Markham and B. C. Sanders. Graph states for quantum secret sharing. *Physical Review A*, 78(4), 2008.

[37] H. P. Nautrup, N. Friis, and H. J. Briegel. Fault-tolerant interface between quantum memories and quantum processors. *Nature Communications*, 8(1):1321, 2017.

[38] M. A. Nielsen. Cluster-state quantum computation. *Reports on Mathematical Physics*, 57(1):147–161, 2006.

[39] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2010.

[40] S. I. Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory. Series B*, 95(1):79–100, 2005.

[41] S.-i. Oum and P. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[42] R. Raussendorf and H. J. Briegel. A One-Way Quantum Computer. *Physical Review Letters*, 86(22):5188–5191, 2001.

[43] J. Ribeiro, G. Murta, and S. Wehner. Fully device-independent conference key agreement. *Physical Review A*, 97(2):022307, 2018.

[44] N. Robertson and P. D. Seymour. Graph minors. I - XXIII. *Journal of Combinatorial Theory, Series B*.

[45] D. Schlingemann and R. F. Werner. Quantum error-correcting codes associated with graphs. *Physical Review A*, 65(1):8, 2002.

[46] J. Spinrad. Recognition of Circle Graphs, 1994.

[47] W. A. Stein and Others. *{S}age {M}athematics {S}oftware ({V}ersion 8.2)*. The Sage Development Team, 2018.

[48] M. Van den Nest, J. Dehaene, and B. De Moor. Efficient algorithm to recognize the local Clifford equivalence of graph states. *Physical Review A*, 70(3):034302, 2004.

[49] M. Van den Nest, J. Dehaene, and B. De Moor. Graphical description of the action of local Clifford transformations on graph states. *Physical Review A*, 69(2):022316, 2004.

[50] M. Van den Nest, W. Dür, G. Vidal, and H. J. Briegel. Classical simulation versus universality in measurement-based quantum computation. *Physical Review A*, 75(1):012337, 2007.

[51] B. Zeng, H. Chung, A. W. Cross, and I. L. Chuang. Local unitary versus local Clifford equivalence of stabilizer and graph states. *Physical Review A*, 75(3), 2007.