



Article submitted to journal

Subject Areas:

Quantum Information Theory

Keywords:

graph states, single-qubit Clifford operations, single-qubit Pauli measurements, computational complexity, rank-width

Author for correspondence:

Axel Dahlberg

e-mail: e.a.dahlberg@tudelft.nlTransforming graph states
using single-qubit operationsAxel Dahlberg¹ and Stephanie Wehner¹¹QuTech, Lorentzweg 1, 2628 CJ Delft, Netherlands

Stabilizer states form an important class of states in quantum information, and are of central importance in quantum error correction. Here, we provide an algorithm for deciding whether one stabilizer (*target*) state can be obtained from another stabilizer (*source*) state by single-qubit Clifford operations (LC), single-qubit Pauli measurements (LPM), and classical communication (CC) between sites holding the individual qubits. What's more, we provide a recipe to obtain the sequence of LC + LPM + CC operations which prepare the desired target state from the source state, and show how these operations can be applied in parallel to reach the target state in constant time. Our algorithm has applications in quantum networks, quantum computing, and can also serve as a design tool - for example, to find transformations between quantum error correcting codes. We provide a software implementation of our algorithm that makes this tool easier to apply.

A key insight leading to our algorithm is to show that the problem is equivalent to one in graph theory, which is to decide whether some graph G' is a *vertex-minor* of another graph G . The vertex-minor problem is in general NP-Complete, but can be solved efficiently on graphs which are not too complex. A measure of the complexity of a graph is the *rank-width* which equals the *Schmidt-rank width* of a subclass of stabilizer states called graph states, and thus intuitively is a measure of entanglement. Here we show that the vertex-minor problem can be solved in time $O(|G|^3)$ where $|G|$ is the size of the graph G , whenever the rank-width of G and the size of G' are bounded. Our algorithm is based on techniques by Courcelle for solving fixed parameter tractable problems, where here the relevant fixed parameter is the rank width. The second half of this paper serves as an accessible but far from exhausting introduction to these concepts, that could be useful for many other problems in quantum information.

1. Introduction

Stabilizer states form a well studied class of quantum states with many applications in quantum information theory and technology. One of the important features of stabilizer states is that they can be described efficiently. As a consequence, quantum circuits consisting only of Clifford operations, and thus mapping stabilizer states to stabilizer states, can be simulated efficiently on a classical computer [18]. Despite this, the class of stabilizer states is still rich enough to be useful in almost any application of quantum networks. An example of such a state that features in many quantum protocols is the GHZ state. In networks, applications thus include quantum secret sharing [26], anonymous transfer [10], conference key agreement [29], and clock synchronization [23]. Furthermore, stabilizer states are also important for quantum computation since they form a universal resource for measurement-based quantum computation [28]. The set of stabilizer states includes cluster states and logical states (codewords) of most quantum error correcting codes [17].

Here, we provide an algorithm for deciding whether a specific multipartite entangled target state can be obtained from a given source state, by local Clifford operations and Pauli measurements on the individual qubits, and classical communication between the sites holding these qubits. More specifically, we consider the problem of deciding whether some stabilizer (*target*) state $|S_t\rangle$ can be produced from some given stabilizer (*source*) state $|S_s\rangle$ by single-qubit Clifford operations, single-qubit Pauli measurement and classical communication: LC + LPM + CC. An efficient algorithm for solving this task has many interesting applications, including:

- Given a source state in a quantum network, or distributed quantum processor, our algorithm allows us to quickly decide if, and how, simple LC + LPM + CC operations can turn this existing source state into a desired target state required by a quantum protocol. This provides a useful tool for entanglement routing and management in a quantum internet, or a networked quantum computer, where fast decisions are essential to combat the limited lifetime of quantum memories.
- Quantum error correcting codes play an important role in realizing fault tolerant quantum computing systems, and have also been put forward as a means to eventually realize all photonic quantum repeaters in the future [5]. Here, our algorithm can be used a design tool to find the measurements to realize a specific target state from a given source state.

We will phrase our results in terms of graph states, which enables us to apply existing algorithmic methods more easily. Graph states form a strict subclass of stabilizer states, which are described by simple undirected unweighted graphs. The vertices in the graph represent qubits initialized in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and edges represent controlled phase gates between the corresponding qubits. Any stabilizer state is in fact single-qubit Clifford equivalent to some graph state [32]. Furthermore, a graph state that is single-qubit Clifford equivalent to a given stabilizer state on n qubits can be found efficiently in time $\mathcal{O}(n^3)$. Thus, if we find an efficient algorithm that decides if some graph state $|G_t\rangle$ can be reached from $|G_s\rangle$ by LC + LPM + CC we also have an efficient algorithm for the more general case where the target and source states are stabilizer states.

Let us now phrase our problem in a way that forms a natural relation to graph properties. To this end, we introduce the notion of a qubit-minor in the following definition, which precisely captures whether a graph state can be reached from another by LC + LPM + CC.

Definition 1.1 (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.

$$|G\rangle \xrightarrow[\text{LPM+CC}]{\text{LC}} |G'\rangle \otimes |\text{junk}\rangle_{V \setminus U}. \quad (1.1)$$

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

$$|G'\rangle < |G\rangle. \quad (1.2)$$

The reason for calling $|G'\rangle$ a qubit-minor of $|G\rangle$ is that this question is in fact equivalent to the graph theoretical question of whether G' is a vertex-minor of G , as we show in theorem 2.2. In the graph theoretical picture, single-qubit Cliffords operations will be replaced by operations called *local complementations* on the graph and single-qubit Pauli measurements by local complementations and vertex-deletions. A vertex-minor of some graph is by definition a graph that can be reached by some sequence of local complementations and vertex-deletions. Equivalence of graphs under local complementations has been studied by Bouchet in [8], which was used by Van den Nest et al. in [31] to find an efficient algorithm to decide whether two graph states are equivalent under single-qubit Clifford operations.

The computational complexity of deciding if a graph G' is a vertex-minor of G , and therefore if $|G'\rangle$ is a qubit-minor of $|G\rangle$, was, to the authors' knowledge, previously unknown. In another paper [14] we show that this decision problem is in fact NP-Complete . There is therefore no efficient algorithm that solves this question in general, unless $\mathbb{P} = \text{NP}$.

(a) Results

Here, we show that the same problem can be solved in cubic time in the number of qubits of $|G\rangle$ on instances where the *Schmidt-rank width* of $|G\rangle$ and the number of qubits of $|G'\rangle$ are bounded¹. This is our first main result which we formally state in theorem 1.1 and prove in section 5(c).

Theorem 1.1. *There exists an algorithm that decides if $|G'\rangle$ is a qubit-minor of $|G\rangle$, and therefore if G' is a vertex-minor of G , and has running time*

$$\mathcal{O}(f(|G'|, r) \cdot |G|^3), \quad (1.3)$$

where r is the rank-width of G which is equal to the Schmidt-rank width of $|G\rangle$, $|G|$ denotes the number of vertices in the graph G and f is some computable function. If conjecture 5.1 is true then there exists an algorithm to the same problem but with running time

$$\mathcal{O}(f(r) \cdot |G|^3). \quad (1.4)$$

Our second main result concerns GHZ-states, which are useful for many applications on a quantum network and is therefore an important target state. A GHZ-state on the qubits in the set U is given as

$$|\text{GHZ}\rangle_U = \frac{1}{\sqrt{2}} \left(\bigotimes_{v \in U} |0\rangle_v + \bigotimes_{v \in U} |1\rangle_v \right). \quad (1.5)$$

One can easily check that the state $|\text{GHZ}\rangle_U$ is single-qubit Clifford equivalent to the graph state $|K_U\rangle$, where K_U is the complete graph with vertex-set U . It is therefore the case that $|\text{GHZ}\rangle_U$ can be mapped from $|G\rangle$ by LC + LPM + CC if and only if $|K_U\rangle$ is a qubit-minor of $|G\rangle$. We show that this question can be solved efficiently if the Schmidt-rank width of $|G\rangle$ is bounded, as captured in theorem 1.2 and proven in section 5(c).

¹Note that the time-dependence on the size of G' can be removed if conjecture 5.1 is true.

Theorem 1.2. *There exists an algorithm that decides if $|K_U\rangle$ is a qubit-minor of $|G\rangle$, and therefore if K_U is a vertex-minor of G and has running time*

$$\mathcal{O}(f(r) \cdot |G|^3), \quad (1.6)$$

where r is the rank-width of G which is equal to the Schmidt-rank width of $|G\rangle$, $|G|$ denotes the number of vertices in the graph G and f is some computable function.

Note in particular that the running time in theorem 1.2 does not depend on U , even if conjecture 5.1 is false. Similarly to theorem 1.2 one can also decide if a graph state has a qubit-minor on a subset U with a given property², efficiently on graph states with bounded Schmidt-rank width, see theorem 5.3.

Both of the two main results, theorem 1.1 and theorem 1.2, rely on a variant of Courcelle's theorem, which we describe more in detail in section 5. Courcelle's theorem states that a large class of graph problems are fixed-parameter tractable. This means that there exist algorithms for these problems which are efficient in the size of the input graphs, provided a certain parameter of these graphs is bounded. This is a very powerful theorem, but a direct implementation of the algorithm given by Courcelle's theorem is not useful in practice. The reason being that even though the algorithm is efficient, the hidden constant factor of the algorithm's asymptotic runtime is huge. This huge constant factor is unavoidable since the theorem is so general and captures many NP -Complete problems. On the other hand, by knowing that a problem can be efficiently solved, one can usually find a more tailored efficient algorithm for the problem at hand, that does not have a huge hidden constant in the runtime. In another paper [14], we provide an efficient algorithm without a huge hidden constant for the problem of deciding whether $|K_U\rangle$ is a qubit-minor of some graph $|G\rangle$, if $|G\rangle$ has Schmidt-rank width one. There are also many other approaches to find practical algorithms for problems captured by Courcelle's theorem, see for example [24] or [16].

We have implemented many of the concepts and algorithms mentioned in this paper in SAGE [3] and MONA [2]. Both the code in SAGE and MONA can be freely accessed from the git-repository at [1]. The functionalities provided by this repository include:

- A function taking two graphs, G and G' , as input and returns TRUE if the graph states $|G\rangle$ and $|G'\rangle$ are equivalent under single-qubit Clifford operations and otherwise returns FALSE. The function has a runtime of $\mathcal{O}(|G|^4)$ and is an implementation of the algorithm described in [8,31].
- A function taking two graphs, G and G' , as input and returns a sequence of operations that takes $|G\rangle$ to a graph state which is single-qubit Clifford equivalent to $|G'\rangle$, if $|G'\rangle < |G\rangle$ and otherwise returns FALSE. This function uses a more sophisticated version of the non-efficient algorithm described in section 23.
- A function taking a graph G and a set U as input and either returns a sequence of operations that takes $|G\rangle$ to a graph state which is single-qubit Clifford equivalent to $|K_U\rangle$ and therefore $|\text{GHZ}\rangle_U$ or returns FALSE. If the function returns FALSE and G has rank-width one, then $|K_U\rangle \not< |G\rangle$ as we show on [14]. The runtime of this function is $\mathcal{O}(|U||G|^3)$.
- As described in section 5 one can express whether $|G'\rangle < |G\rangle$ in a logic called monadic second-order logic. We have implemented the expression for $|G'\rangle < |G\rangle$ in MONA which is a software to translate such logic expressions to finite-state automata. This can then be used to construct efficient algorithms for graphs of bounded rank-width.

(b) Overview

In section 2 we describe graph states and introduce the notions of qubit-minors and vertex-minors. We also provide a non-efficient but correct algorithm for deciding the vertex-minor

²Expressible in C_2MS .

problem, taking any graph as input, in section 3. Note that such an algorithm is necessarily non-efficient, unless $\mathbb{P} = \text{NP}$, since the problem it solves is in general NP-Complete . Furthermore, we describe how the corresponding operations on the graph states can be applied in constant time in section 4. In section 5 we provide an efficient algorithm for graphs with bounded rank-width by making use of monadic second-order logic and Courcelle's theorem. It is our intention that this section can also be used as a short introduction for those not familiar with these concepts.

2. Background

(a) Notation

All graphs in this paper are simple, unweighted and undirected, where simple means that there are no self-loops or multi-edges. Given a graph $G = (V, E)$, we will sometimes denote the vertex-set as $V(G) = V$ and the edge-set as $E(G) = E$. By the size of a graph we mean the number of vertices, which we denote as $|G| = |V(G)|$. We will denote the neighborhood of a vertex as

$$N_v^{(G)} = \{u \in V(G) : (v, u) \in E(G)\}. \quad (2.1)$$

If it is clear which graph is considered, we will also sometimes write N_v . The induced subgraph of G on the subset $U \subseteq V(G)$ is denoted as $G[U]$ and is the graph with vertex-set U and edge-set

$$\{(u, v) \in E(G) : u \in U \wedge v \in U\}. \quad (2.2)$$

We denote vertex-deletions by $\setminus v$ such that $G \setminus v = G[V(G) \setminus \{v\}]$.

The Pauli matrices will be denoted as

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (2.3)$$

The single-qubit Clifford group \mathcal{C} is the normalizer of the Pauli group $\mathcal{P} = \langle i\mathbb{I}, X, Z \rangle$, i.e.

$$\mathcal{C} = \{C \in \mathcal{U} : (\forall P \in \mathcal{P} : CPC^\dagger \in \mathcal{P})\}, \quad (2.4)$$

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

Assume that v_i is the label of a qubit which is part of some multi-qubit state $|\psi\rangle_{v_1 \dots v_i \dots v_n}$. We will then denote P_{v_i} as the operation

$$P_{v_i}^{(v_1 \dots v_n)} = (\mathbb{I})_{v_1} \otimes \dots \otimes (P)_{v_i} \otimes \dots \otimes (\mathbb{I})_{v_n}, \quad (2.5)$$

where $P \in \{\mathbb{I}, X, Y, Z\}$. We will never write an explicit ordering of the qubits in a multi-qubit state $|\psi\rangle_{v_1 \dots v_i \dots v_n}$ and rather write $|\psi\rangle_V$, where V is the set $\{v_1, \dots, v_n\}$. For explicit calculations one just needs to use a consistent ordering. Similarly for the operation $P_{v_i}^{(v_1 \dots v_n)}$ we will write $P_{v_i}^V$ or even P_{v_i} when it is clear which set V is considered.

(b) Graph states

A graph state $|G\rangle$ is a quantum state described by a simple unweighted undirected graph G , where the vertices of G correspond to the qubits of $|G\rangle$. Formally, let $G = (V, E)$ be a graph, the graph state $|G\rangle$ is then defined as

$$|G\rangle = \prod_{e \in E} C_Z^e \bigotimes_{v \in V} |+\rangle_v, \quad (2.6)$$

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

$$C_Z^{(u,v)} = |0\rangle\langle 0|_u \otimes \mathbb{I}_v^{(V \setminus u)} + |1\rangle\langle 1|_u \otimes Z_v^{(V \setminus u)}. \quad (2.7)$$

A graph state is also a stabilizer state [20]. The generators of the stabilizer group of $|G\rangle$ can be written as

$$g_v = X_v \prod_{u \in N_v} Z_u. \quad (2.8)$$

In fact, any stabilizer state can be made into some graph state by only performing single-qubit Clifford operations [32]. If two states, $|\psi\rangle$ and $|\phi\rangle$ are related by some sequence of single-qubit Clifford operations, we denote this by $|\psi\rangle \sim_{\text{LC}} |\phi\rangle$. Given a stabilizer state $|S\rangle$, one can find a graph state $|G\rangle$ such that $|S\rangle_V \sim_{\text{LC}} |G\rangle_V$, by simply performing Gaussian elimination followed by certain operations on the columns of a matrix which rows are the symplectic form of the generators of S [20].

To study what graph states can be reached from a given graph state we introduce the notion of a qubit-minor as defined in definition 1.1. It turns out that the question of whether $|G'\rangle$ is a qubit-minor of $|G\rangle$ is equivalent to whether G' is a vertex-minor of G , as we describe below.

(c) Local Clifford operations

Let's consider the following sequence of single-qubit Clifford operations

$$U_v^{(G)} = \exp\left(-i\frac{\pi}{4}X_v\right) \prod_{u \in N_v} \exp\left(i\frac{\pi}{4}Z_u\right). \quad (2.9)$$

As shown in [20], the operation $U_v^{(G)}$ on the state $|G\rangle$ can be seen as an operation on the graph G since

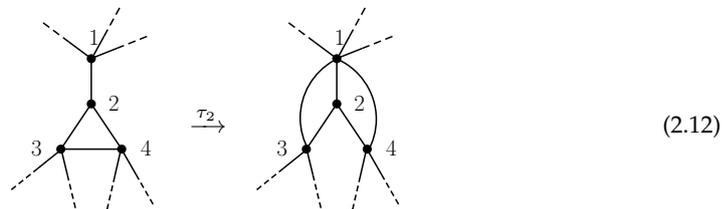
$$U_v^{(G)}|G\rangle = |\tau_v(G)\rangle \quad (2.10)$$

where τ_v is a local complementation on the vertex v , as defined in definition 2.1 and illustrated in equation (2.12).

Definition 2.1 (local complementation). *A local complementation τ_v acts on a vertex v of a graph G by complementing the induced subgraph on the neighborhood of v . The neighborhoods of the graph $\tau_v(G)$ are therefore given by*

$$N_u^{\tau_v(G)} = \begin{cases} N_u \Delta (N_v \setminus \{u\}) & \text{if } (u, v) \in E(G) \\ N_u & \text{else} \end{cases}, \quad (2.11)$$

where Δ denotes the symmetric difference between two sets.



Surprisingly, any single-qubit Clifford operation which takes some graph state to another graph state can be seen as some sequence of local complementations on the corresponding graphs. This was proven in [32] and we restate this theorem here.

Theorem 2.1 (Van den Nest [32]). *Two graph states $|G\rangle$ and $|G'\rangle$ are equivalent under single-qubit Clifford operations if and only if their corresponding graphs G and G' are related by some sequence of local complementations.*

Let $m = v_1 v_2 \dots v_l$ be a sequence of vertices of G , then we denote the sequence of local complementations on the vertices in m as

$$\tau_m(G) = \tau_{v_l} \circ \dots \circ \tau_{v_2} \circ \tau_{v_1}(G). \quad (2.13)$$

If there exists a sequence m such that $\tau_m(G) = G'$ then we write this as $G \sim_{\text{LC}} G'$. Theorem 2.1 can therefore be stated as

$$|G\rangle \sim_{\text{LC}} |G'\rangle \Leftrightarrow G \sim_{\text{LC}} G'. \quad (2.14)$$

Testing 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 [8].

(d) Local Pauli measurements

How should the corresponding graph of a graph state be updated when a measurement is performed? In [21] it is shown how the Pauli projectors $P_v^{(X,\pm)}$, $P_v^{(Y,\pm)}$, $P_v^{(Z,\pm)}$ act on graph states³

$$P_v^{(Z,\pm)}|G\rangle = \frac{1}{2}|Z, \pm\rangle_v \otimes U_v^{(Z,\pm)}|G \setminus v\rangle \quad (2.15)$$

$$P_v^{(Y,\pm)}|G\rangle = \frac{1}{2}|Y, \pm\rangle_v \otimes U_v^{(Y,\pm)}|\tau_v(G) \setminus v\rangle \quad (2.16)$$

$$P_v^{(X,\pm)}|G\rangle = \frac{1}{2}|X, \pm\rangle_v \otimes U_e^{(X,\pm)}|T_e(G) \setminus v\rangle \quad \text{if } |N_v| > 0 \quad (2.17)$$

where e is an edge of G incident on the vertex v . Choosing a different edge e' incident on v gives a single-qubit Clifford-equivalent graph state, i.e. $|T_{e'}(G) \setminus v\rangle \sim_{\text{LC}} |T_e(G) \setminus v\rangle$. The operation $T_{(u,v)}$ is called a pivot and is defined as $T_{(u,v)} = \tau_v \circ \tau_u \circ \tau_v$. The pivot can simply be specified by an undirected edge since

$$\tau_v \circ \tau_u \circ \tau_v(G) = \tau_u \circ \tau_v \circ \tau_u(G) \quad \text{if } (u, v) \in E(G) \quad (2.18)$$

as shown in [7]. The operators $U_v^{(P,\pm)}$ are sequences of single-qubit Clifford operations and take the post-measurement state to a graph state. The exact form of these correction operators can be found in [21] but we will only need the ones for measurements in the standard (Z -) basis, which are given by

$$U_v^{(Z,+)} = \mathbb{I}_v, \quad U_v^{(Z,-)} = \prod_{u \in N_v} Z_u. \quad (2.19)$$

Since the correction operators are sequences of single-qubit Clifford operations it is therefore the case that

$$|G \setminus v\rangle, \quad |\tau_v(G) \setminus v\rangle, \quad |T_e(G) \setminus v\rangle \quad (2.20)$$

are all qubit-minors of $|G\rangle$. Furthermore, Bouchet proved the following.

Lemma 2.1 (Bouchet, (9.2) in [7]). *If $G \sim_{\text{LC}} G'$ then $G' \setminus v$ is LC-equivalent to $G \setminus v$, $\tau_v(G) \setminus v$ or $T_e(G) \setminus v$, where e is some fixed edge incident on v in G .*

We therefore have the following lemma.

Lemma 2.2. *Let $|G\rangle$ be a graph state, $v \in V(G)$ be a vertex and $e \in E(G)$ be an edge incident on v . Furthermore, assume that $|G'\rangle$ is a qubit-minor of $|G\rangle$, where $V(G') = V(G) \setminus v$ and that G' has no vertices of degree zero. Then $|G'\rangle$ is single-qubit Clifford-equivalent to at least one of the three states in equation (2.20).*

³For the special case when $|N_v| = 0$, a measurement in the X -basis does not change the graph state since this is then $|G\rangle = |+\rangle_v \otimes |G \setminus v\rangle$.

Proof. Since $|G'\rangle$ is a qubit-minor of $|G\rangle$ we know that there exists a sequence \mathcal{W} of single-qubit Clifford operations, single-qubit Pauli measurements and classical communication that takes $|G\rangle$ to $|G'\rangle$, by definition. Any single-qubit Pauli measurement on a qubit u gives a product states between qubit u and the rest of the qubits, see equations (2.15)-(2.17). The sequence \mathcal{W} cannot therefore contain a single-qubit Pauli measurement on a qubit u , different from v , since u has non-zero degree in the graph G' by assumption. Without loss of generality we can in fact assume that \mathcal{W} is a sequence of single-qubit Clifford operations followed by a measurement of v in the standard basis and then by another sequence of single-qubit Clifford operations. The reason why it is sufficient to consider a measurement in the standard basis is that any other Pauli measurement can be simulated by performing some single-qubit Clifford operation followed by a measurement in the standard basis. Assume that the sequence of single-qubit Clifford operations before (after) the measurement is described by the sequences of local complementations m (m'), which exists due to theorem 2.1. We therefore have that

$$\tau_{m'}(\tau_m(G) \setminus v) = G'. \quad (2.21)$$

Using lemma 2.1 we have that $\tau_m(G) \setminus v$, and therefore G' , is LC-equivalent to either $G \setminus v$, $\tau_v(G) \setminus v$ or $T_e(G) \setminus v$. Finally, by theorem 2.1 the lemma follows. \square

(e) Vertex-minors

As mentioned, the question of whether $|G'\rangle$ is a qubit-minor of $|G\rangle$ is equivalent to whether G' is a vertex-minor of G . Vertex-minors were introduced by Bouchet in [7] but by the name of l -reductions.

Definition 2.2 (vertex-minor). *Let G be a graph. G' is called a vertex-minor of G if it can be reached by some sequence of local complementations and vertex-deletions. Equivalently, G' is called a vertex-minor of G if there exists a sequence of vertices m such that*

$$\tau_m(G)[V(G')] = G' \quad (2.22)$$

If G' is a vertex-minor of G , we denote this as

$$G' < G \quad (2.23)$$

In the previous sections we have seen that single-qubit Clifford operations that take graph states to graph states can be seen as local complementations on the corresponding graph and similarly for single-qubit Pauli measurements and vertex-deletions. The relation between qubit-minors and vertex-minors is captured by the following theorem.

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

$$|G'\rangle < |G\rangle \Leftrightarrow G' < G. \quad (2.24)$$

Proof. Assume first that $|G'\rangle$ is a qubit-minor of $|G\rangle$. By the same arguments as in the proof of lemma 2.2 we then have that there exists a sequence of vertices m such that

$$\tau_m(G)[V(G')] = G' \quad (2.25)$$

and by definition that G' is a vertex-minor of G . Assume now on the other hand that G' is a vertex-minor of G , i.e. that $\tau_m(G)[V(G')] = G'$ for some m . We can then go from $|G\rangle$ to $|G'\rangle$ by simply performing the single-qubit Clifford operations corresponding to the sequence of local complementation specified by m and then measure the qubits $V(G) \setminus V(G')$ in the standard basis. If the correct corrections, i.e. $U_v^{(Z, \pm)}$, are applied after the measurements, the state $|G'\rangle$ is reached. \square

So to check whether a graph state has a certain qubit-minor we can check if the corresponding graph has a certain vertex-minor. 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 U . We then have that

$$|G'| < |G| \Leftrightarrow G'[V(G) \setminus U] < G. \quad (2.26)$$

(f) Rank-width

In this section we introduce the notion of rank-width, which is a complexity measure of a graph. It is in some ways similar to the tree-width, introduced in [30]. The tree-width captures essentially how tree-like the graph is. This is useful for finding algorithms for problems on graphs of bounded tree-width, motivated by the fact that many graph problems are easy on trees. More on algorithms for problems on graphs of bounded tree-width can be found in [15]. Rank-width, compared to tree-width, captures a larger class of graphs with similar complexity. For example, the complete graph has very low complexity, due to its highly symmetric nature, but the tree-width is in this case maximal. On the other hand the rank-width is one for both trees and complete graphs. In fact, it turns out that the graphs of rank-width one are exactly the distance-hereditary graphs, see [27].

We start by defining the cut-rank of a graph. To do this we will use the following notation for a graph G with vertices V and adjacency matrix Γ and two subsets of the vertices $A, B \subseteq V$; $\Gamma[A, B]$ is the $|A| \times |B|$ -matrix describing the connections between the sets A and B . So, for $a \in A$ and $b \in B$, the element $(\Gamma[A, B])_{ab}$ is 1 if (a, b) is an edge in G and 0 otherwise.

Definition 2.3 (cut-rank). *Let's assume that A is a subset of the vertices V of some graph G with adjacency matrix Γ . The cut-rank $\text{cutrk}_A(G)$ of G with respect to A , is then defined as*

$$\text{cutrk}_A(G) \equiv \text{rank}_{\mathbb{F}_2}(\Gamma[A, V \setminus A]), \quad (2.27)$$

where $\text{rank}_{\mathbb{F}_2}$ is the rank over the finite field of order two.

Note that the cut-rank is symmetric in the sense that

$$\text{cutrk}_A(G) = \text{rank}_{\mathbb{F}_2}(\Gamma[A, V \setminus A]) = \text{rank}_{\mathbb{F}_2}(\Gamma[V \setminus A, A]^\top) = \text{cutrk}_{V \setminus A}(G). \quad (2.28)$$

Interestingly the cut-rank with respect to A of a graph G is in fact equal to the Schmidt-rank of the state $|G\rangle$ with respect to the bipartition $(A, V \setminus A)$.⁴

Next we define what is called a rank-decomposition of a graph.

Definition 2.4 (rank-decomposition). *A rank-decomposition of a graph G is a pair $\mathcal{R} = (\mathcal{T}, \mu)$, where \mathcal{T} is a subcubic tree and μ is a bijection $\mu: V(G) \rightarrow \{l : l \text{ is a leaf of } \mathcal{T}\}$. A subcubic tree is a tree with at least two vertices and each vertex has degree less or equal to 3. Any edge e in \mathcal{T} splits the tree into two connected components upon deletion and therefore induces a partition (A_e, B_e) of the leaves. The width of an edge e of the subcubic tree is defined as the cut-rank of the corresponding partition. Furthermore the width of the rank-decomposition is defined as the maximum width over all edges, i.e.*

$$\text{width}_{\mathcal{R}}(G) \equiv \max_{e \in E(\mathcal{T})} \text{cutrk}_{\mu^{-1}(A_e)}(G). \quad (2.29)$$

To simplify notation we write the cut-rank induced by a rank-decomposition (\mathcal{T}, μ) and an edge e as

$$\text{cutrk}_{\mu^{-1}(\mathcal{T}, e)}(G) \equiv \text{cutrk}_{\mu^{-1}(A_e)}(G). \quad (2.30)$$

This allows us to define the rank-width of a graph.

⁴This is proven in [20], see proposition 10.

Definition 2.5 (rank-width). *The rank-width $\text{rwd}(G)$ of a graph G is the minimum width over all rank-decompositions, i.e.*

$$\text{rwd}(G) \equiv \min_{\mathcal{R}} \text{width}_{\mathcal{R}}(G) = \min_{(\mathcal{T}, \mu)} \max_{e \in E(\mathcal{T})} \text{cutrk}_{\mu^{-1}(\mathcal{T}, e)}(G). \quad (2.31)$$

The rank-width of the graph G is related to the entanglement of the state $|G\rangle$, although as a relatively unknown entanglement monotone. In [33] the corresponding entanglement monotone is called the *Schmidt-rank width* and is defined for general quantum states. For graph states, the Schmidt-rank width of the state and the rank-width of the corresponding graph coincide. There they also give an interpretation of the Schmidt-rank width as a quantifier for the optimal description of the state using a tree tensor network.

3. A non-efficient but general algorithm

Here we describe an algorithm that decides if G' is a vertex-minor of G and returns a sequence of local complementations m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$ if such a sequence exists. This algorithm works for any G and G' and has a running time of $\mathcal{O}(3^{n-k}(k^4 + (n-k)n^2))$, where $k = |G'|$ and $n = |G|$. Obviously this algorithm is not efficient, due to the exponential scaling in the size-difference of the graphs, but is still useful for smaller graphs or when $n - k$ is bounded. For finite n , the algorithm described in this section is also a useful benchmark for the efficient algorithm described in section 5 for graphs of bounded rank-width. To prove that the algorithm is correct, we first prove theorem 3.1.

Algorithm 1 Non-efficient algorithm that decides if $G' < G$.

Input: (G, G') .

Output: A sequence m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$ if $G' < G$.
 \perp if $G' \not< G$.

```

1: function IS_VM( $G, G'$ )
2:   if  $V(G') \not\subseteq V(G)$  then
3:     return  $\perp$ 
4:   end if
5:   if  $V(G) = V(G')$  then
6:     if  $G \sim_{\text{LC}} G'$  then
7:       return [] ▷ Return an empty sequence
8:     else
9:       return  $\perp$ 
10:    end if
11:  else
12:    Let  $v$  be a vertex in  $V(G) \setminus V(G')$ 
13:    Let  $m_Z = \text{IS\_VM}(G \setminus v)$ 
14:    if  $m_Z \neq \perp$  then
15:      return  $m_Z$ 
16:    else
17:      Let  $m_Y = \text{IS\_VM}(\tau_v(G) \setminus v)$ 
18:      if  $m_Y \neq \perp$  then
19:        return  $m_Y \parallel [v]$  ▷ Concatenate  $[v]$  to  $m_Y$  and return.
20:      else
21:        Let  $u$  be a vertex incident to  $v$  in  $G$ 
22:        Let  $m_X = \text{IS\_VM}(T_{(v,u)}(G) \setminus v)$ 
23:        if  $m_X \neq \perp$  then
24:          return  $m_X \parallel [u, v, u]$  ▷ Concatenate  $[u, v, u]$  to  $m_X$  and return.
25:        else
26:          return  $\perp$ 
27:        end if
28:      end if
29:    end if
30:  end if
31: end function

```

As seen in the previous section, when making a Pauli X measurement on a qubit v , the choice of the edge e incident on v gives different graph states that are single-qubit Clifford equivalent to the post-measurement state. To simplify the results of this section we introduce the following graph operation which allows us to not have to deal with different choices of edges, incident on the qubit being measured.

Definition 3.1. Let v be a vertex in the graph G . We define T_v as the following operation

$$T_v(G) = \begin{cases} T_{e_v}(G) & \text{if } |N_v| > 0 \\ G & \text{if } |N_v| = 0 \end{cases} \quad (3.1)$$

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 $T_v(G)$.

We are now ready to prove a generalization of lemma 2.2, which algorithm 1 is built on.

Theorem 3.1. Let G and G' be two graphs and U be the set $V(G) \setminus V(G') = \{v_1, \dots, v_{n-k}\}$. Furthermore, let \mathcal{P}_U denote the set of graph operations

$$\mathcal{P}_U = \{P_{v_{n-k}} \circ \dots \circ P_{v_1} : P_v \in \{(-) \setminus v, \tau_v(-) \setminus v, T_v(-) \setminus v\}\} \quad (3.2)$$

Then we have that

$$G' < G \Leftrightarrow \exists P \in \mathcal{P}_U : G' \sim_{\text{LC}} P(G). \quad (3.3)$$

Proof. If there exists a P in \mathcal{P}_U such that $G' \sim_{\text{LC}} P(G)$ then we clearly have that $G' < G$, since any such P is some sequence of local complementations and vertex-deletions. Assume now that $G' < G$. We will prove by induction on $n - k$ that there exists a P in \mathcal{P}_U such that $G' \sim_{\text{LC}} P(G)$. For $n - k = 1$ this follows directly from lemma 2.1. Assume therefore that it is true for $n - k = l$. We now show that this implies that it is also true for $n - k = l + 1$. Since $G' < G$ we know that $\tau_m(G)[V(G')] = G'$ for some m . Let v be a vertex in $V(G')$ and consider the graph $\tilde{G} = \tau_m(G)[V(G') \cup \{v\}]$. Note that $G' = \tilde{G} \setminus v$. Clearly we have that $G' < \tilde{G} < G$ and by the induction assumption we know that

$$\exists P \in \mathcal{P}_{U \setminus \{v\}} : \tilde{G} \sim_{\text{LC}} P(G). \quad (3.4)$$

Then from lemma 2.1 we know that $G' = \tilde{G} \setminus v$ is LC-equivalent to at least one of the following graphs

$$P(G) \setminus v, \quad \tau_v(P(G)) \setminus v, \quad T_v(P(G)) \setminus v \quad (3.5)$$

and the theorem follows. \square

From theorem 3.1 we see that to check if G' is a vertex-minor of G it is sufficient to check if one of the graphs in

$$\left\{ P(G) : P \in \mathcal{P}_{V(G) \setminus V(G')} \right\} \quad (3.6)$$

are LC-equivalent to G' . Note that there are possibly $3^{|V(G) \setminus V(G')|}$ graphs in the set in equation (3.6) to check. As mentioned earlier it is possible to check whether two graphs are LC-equivalent in time $O(k^4)$, where k is the size of the graphs. The explicit algorithm for checking if G' is a vertex-minor of G is stated in algorithm 1 and theorem 3.2 captures the proof that it is correct and what its running time is.

Theorem 3.2. Algorithm 1 returns a sequence m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$ if $G' < G$ and returns \perp if $G' \not< G$. Furthermore the runtime is $\mathcal{O}(3^{n-k}(k^4 + (n - k)n^2))$, where $k = |G'|$ and $n = |G|$.

Proof. We first prove that the algorithm is correct. Since the algorithm calls itself recursively with the three graphs as in in line 13, 17 and 21 it will generate all the graphs in equation (3.6). That is, the graphs tested for LC-equivalence against G' in line 6 are exactly the graphs in equation (3.6). Furthermore, if at least one of the base-cases, i.e. line 5-10, return an empty sequence then the top-level call to the algorithm will return a sequence m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$. On the other hand, if all of the base-cases returns \perp then the top-level call returns \perp . By theorem 3.1 it follows that the algorithm is correct.

Let's now consider the runtime of the algorithm. We assume that the graphs are given as their adjacency matrices. Let's denote the running time by $T(n, k)$. Picking a vertex v from the set $V(G) \setminus V(G')$, as in line 12, can be done in time $\mathcal{O}(n)$. There are three recursive calls on line 13, 17 and 21, where also four local complementations are performed⁵. Each local complementation can be done in quadratic time in the size of the graph. From the definition of local complementation, definition 2.1, we see that this can be done by adding the row of the adjacency matrix corresponding to the vertex where the local complementation is performed to

⁵Three local complementations for the pivot.

the rows of its neighbors, where adding means vector-addition modulo 2. We therefore have that the running time of the full algorithm has the following recursive expression

$$T(n, k) = 3T(n - 1, k) + \mathcal{O}(n^2) \quad (3.7)$$

where $T(k, k) = \mathcal{O}(k^4)$ from testing LC-equivalence of the graphs as the base-case. By induction we see that the running time can be expressed as

$$T(k + l, k) = 3^l \mathcal{O}(k^4) + \sum_{i=0}^{l-1} 3^i \mathcal{O}((k + l - i)^2) \quad (3.8)$$

Evaluating the above expression for $l = n - k$ we get

$$T(n, k) = 3^{n-k} (\mathcal{O}(k^4) + \sum_{i=0}^{n-k-1} \mathcal{O}((n - i)^2)) \quad (3.9)$$

$$= \mathcal{O}(3^{n-k}(k^4 + (n - k)n^2)) \quad (3.10)$$

□

4. Constant time transformation

So far we have considered the task of finding the operations that take some graph state to its qubit-minor, but what is the best way to apply these operations to the state when they are found? Assume that we have, by some classical (or quantum) algorithm, found a sequence of operations that takes us from the current state $|G\rangle$ to the target state $|G'\rangle$, i.e. $|G'\rangle$ is a qubit-minor of $|G\rangle$. There are different ways to express these operations, for example as a sequence of single-qubit Clifford operations and single-qubit Pauli measurements or as a sequence of local complementations and vertex-deletions on the corresponding graph. From the previous section we have also seen how these different representations can be mapped to each other. Let's therefore assume that we have expressed the sequence of operations as local complementations m followed by vertex-deletions of the vertices in $V(G) \setminus V(G')$. The reason for doing this is that we can now perform the single-qubit Clifford operations corresponding to m in parallel and then simultaneously measure all the qubits in $V(G) \setminus V(G')$ in the standard basis. The simultaneous measurements in the standard basis are possible since the corrections $U_v^{(Z, \pm)}$ are either the identity or Z on the neighbors of v and do therefore not change the measurement basis of neighboring vertices, in contrast to Pauli X and Y measurement.

We still need to know what corrections that are needed, depending on the measurement outcomes of the qubits $V(G) \setminus V(G')$. In appendix A we show that vertex $v \in V(G')$ only need to apply $(Z_v)^{y_v}$, where

$$y_v = \sum_{u \in N_v^{(G)} \setminus V(G')} x_u \pmod{2} \quad (4.1)$$

and $x_u \in \{0, 1\}$ is the measurement outcome of node u . In other words, a Z operations is applied to qubit v if the parity of the measurement outcomes of the neighbors of v (in G) is odd. Otherwise, no correction is applied to qubit v . We emphasize that the corrections of the qubit v only depend on the measurement outcomes of the neighborhood of that qubit in the graph G .

Another advantage of only performing measurements in the standard basis is that in some cases it is possible to extract $|G'\rangle$ without destroying all the rest of the entanglement in the original state. More specifically, consider the vertices that are adjacent to at least one vertex in $V(G')$ but which are not in $V(G')$ themselves. These vertices are exactly the ones in the set

$$N_{V(G')} = \left(\bigcup_{v \in V(G')} N_v^{(\tau_m(G))} \right) \setminus V(G'). \quad (4.2)$$

Assume that $N_{V(G')} \neq V(G) \setminus V(G')$. Then the deletion of all the vertices in $N_{V(G')}$ from the graph $\tau_m(G)$ gives a graph with two connected components $G' = \tau_m(G)[V(G')]$ and $\tau_m(G)[V(G) \setminus (N_{V(G')} \cup V(G'))]$. Let's denote the second connected component G_{rest} . We can then see that if after performing the single-qubit Clifford operations corresponding to m , we only measure the qubits in $N_{V(G')}$ in the standard basis, followed by corrections, we arrive at the following state

$$|G'\rangle \otimes |G_{\text{rest}}\rangle \otimes \bigotimes_{v \in N_{V(G')}} |+\rangle_v. \quad (4.3)$$

The entanglement in $|G_{\text{rest}}\rangle$ is then not wasted. Since there are in fact multiple sequences of vertices m such that $\tau_m(G)[V(G')] = G'$, one can also try to minimize the neighborhood $N_{V(G')}$ and therefore maximize the size of $|G_{\text{rest}}\rangle$.

5. Efficient algorithm based on theorem by Courcelle

As mentioned, we show in another paper [14] that the problem of deciding if G' is a vertex-minor of G is \mathbb{NP} -Complete in general. Fortunately the problem is fixed-parameter tractable in the rank-width of G and in general the size of G' , which follows from results by Oum and Courcelle in [13] as we show below. The statement that a problem is fixed parameter-tractable in some parameter r means that there exists an algorithm that solves the problem and has running time

$$\mathcal{O}(f(r) \cdot p(n)) \quad (5.1)$$

where p is some polynomial and n is the size of the input to the problem. Many \mathbb{NP} -Complete problems are fixed-parameter tractable, which means that their time complexity is not necessarily super-polynomial in the input size but rather in the parameter r . For \mathbb{NP} -Complete fixed-parameter tractable problems, the factor $f(r)$ must scale super-polynomially with n in the worst case, unless $\mathbb{P} = \mathbb{NP}$.

In 1990, Courcelle proved that a large class of graph problems are fixed-parameter tractable in the tree-width of the graph [11]. Courcelle's theorem states that any graph problem specified by a monadic second-order logic (MS) formula can be solved in linear time on graphs of bounded tree-width. The tree-width is a notion which essentially describes how tree-like a graph is [30]. Many problems that are hard in general become tractable on trees, as for example the subgraph isomorphism problem [19]. The same holds for graphs which are not too different from trees, i.e. have a low tree-width, which is exactly what Courcelle's theorem states. Since the original theorem by Courcelle, there has also been many generalizations including the same statement but using rank-width. Bounded rank-width captures a larger class of graph than tree-width, for example complete graphs have minimal rank-width. Importantly here is that rank-width is invariant under local complementations and non-increasing under vertex-deletions [27]. We give more details on rank-width in appendix 2(f).

MS logic is an extension of first-order logic, which allows for quantification over sets [12]. Courcelle's theorem actually holds for a strictly more expressive logic called counting monadic second-order logic (CMS) where one can also express whether the size of a set is zero modulo p [12]. A sublanguage of CMS is $C_2\text{MS}$ where p is restricted to be 2 and one can therefore express whether the size of a set is even or odd.

Any graph problem specified by a $C_2\text{MS}$ formula can be solved in cubic time on graphs of bounded rank-width, which is due to theorem 6.55 in [12]. We state this formally in theorem 5.1. This is the result we make use of in this section to find an efficient algorithm for the vertex-minor problem.

It turns out that the vertex-minor problem is expressible in $C_2\text{MS}$, which we formally state in theorem 5.2. By theorem 5.1 and theorem 5.2 we see that the problem of deciding whether $|G'\rangle$ is a qubit-minor $|G\rangle$ is fixed-parameter tractable in the rank-width of G and in the size of G' in general, as we captured in theorem 1.1. The reason the qubit-minor problem is fixed-parameter tractable in both $\text{rwd}(G)$ and $|G'|$ is because the formula $\text{VM}_{G'}$ in equation (5.20) depends on G' .

Note that if conjecture 5.1 is true this dependence of $|G'|$ in the running time can be removed. If G' is restricted to be a certain type of graph or if we ask the question whether G has a vertex-minor on the subset $U \subseteq V(G)$ with a given property instead, then the running time does not need to depend G' or U respectively, see theorems 1.2 and 5.3, even if conjecture 5.1 is false.

Algorithm 2 below gives a high-level description of how to solve the vertex-minor problem efficiently on graphs of bounded rank-width. The overall runtime of the algorithm is $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$ and is dominated by line 6. In line 5, the C_2MS formula VM' defined in equation (5.40) can be constructed in time $\mathcal{O}(|G'|)$. An assignment α_y as in line 6 can be found in time $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$ due to Courcelle's theorem for CMS selection problems [12] together with a proof similar to theorem 5.1. Finally a sequence of switchings m taking the Eulerian vector $(\emptyset, V(G), \emptyset)$ to the $\alpha_y(X_e, Y_e, Z_e)$ can be done in time $\mathcal{O}(|G|)$ as we show in appendix C.

Algorithm 2 Algorithm that decides if $G' < G$. Runtime: $\mathcal{O}(f(|G'|, r) \cdot |G|^3)$

Input: (G, G') where $\text{rwd}(G) = r$.

Output: A sequence m such that $\tau_m(G)[V(G')] = G'$ if $G' < G$.

\perp if $G' \not< G$.

```

1: function IS_VM( $G, G'$ )
2:   if  $V(G') \not\subseteq V(G)$  then
3:     return  $\perp$ 
4:   end if
5:   Construct the  $C_2MS$  formula  $VM'_{G'}(\mathcal{X}, X_e, Y_e, Z_e)$  ▷ See eq. 5.40.
6:   Find an assignment  $\alpha_y$  such that  $G \models VM'_{G'}(\mathcal{X} \mapsto V(G'), \alpha_y(X_e, Y_e, Z_e))$ 
7:   if There is no such  $\alpha_y$  then
8:     return  $\perp$ 
9:   else
10:    Find a sequence of switchings  $m$  taking  $(\emptyset, V(G), \emptyset)$  to  $\alpha_y(X_e, Y_e, Z_e)$  ▷ See sec. 5(d)
11:    return  $m$ 
12:   end if
13: end function

```

(a) Monadic second-order logic

Monadic second-order logic is a sublanguage of second-order logic which in turn is an extension of first-order logic. In MS one can quantify over sets⁶ compared to first-order logic which is restricted to only quantification over elements. For a detailed reference on MS and its extensions, see the book by Courcelle and Engelfriet [12].

MS formulas on graphs uses variables which are either vertex variables x, y, \dots or set variables X, Y, \dots , which are sets of vertices. A MS formula is a finite string built up by the atomic formulas $x = y, x \in X$ and $\text{ADJ}(x, y)$ ⁷ together with the following recursive rules [24]:

- (i) If ϕ is a formula, then so is $\neg\phi$.
- (ii) If $\phi_1 \dots \phi_l$ are formulas, then so are $(\phi_1 \vee \dots \vee \phi_l)$ and $(\phi_1 \wedge \dots \wedge \phi_l)$.
- (iii) If ϕ is a formula, then so are $\exists x : \phi, \forall x : \phi, \exists X : \phi$ and $\forall X : \phi$.

The variables of a formula which are not part of a quantifier, as in rule (iii) above, are called free variables. A formula with no free variables is called a sentence. We write $\phi(X_1, \dots, X_l, x_1, \dots, x_m)$ for a formula with free variables $X_1 \dots X_l x_1 \dots x_m$. To simplify

⁶In second-order logic one can more generally quantify over predicates. MS is restricted to quantification over predicates with one argument (monadic), which is equivalent to quantification over sets.

⁷Expressing whether (x, y) is an edge in the considered graph.

formulas we will sometimes make use of the following abbreviations

$$(\phi \Rightarrow \psi) \equiv (\neg\phi \vee \psi) \quad (5.2)$$

$$(\phi \Leftrightarrow \psi) \equiv ((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)) \quad (5.3)$$

$$(X \subseteq Y) \equiv (\forall x : (x \in X \Rightarrow x \in Y)). \quad (5.4)$$

A MS formula is related to a graph G by the atomic formula $\text{ADJ}(x, y)$, which is true if and only if (x, y) is an edge in G . If a MS sentence ϕ is true on a graph G , we say that G models ϕ and write this as $G \models \phi$. For a formula with free variables, an assignment α is a mapping from the free variables to vertices and subsets of vertices of a graph G . If a MS formula $\phi(X_1, \dots, X_l, x_1, \dots, x_m)$ is true on a graph G with the assignment α , we say that α satisfies ϕ on G and write this as

$$G \models \phi(\alpha(X_1), \dots, \alpha(X_l), \alpha(x_1), \dots, \alpha(x_m)). \quad (5.5)$$

If α assigns v to the free variable x , we write this as $x \mapsto v$. Furthermore, if a formula has free variables $\mathcal{X} = X_1 \dots X_l x_1 \dots x_m$ then we sometimes write $\phi(\mathcal{X})$ and $\phi(\alpha(\mathcal{X}))$ for an assignment of these variables. When considering a formula on a graph we implicitly assume that the quantifiers are over the vertex-set of the graph, i.e.

$$G \models \forall x : \phi(x) \quad \text{iff} \quad \bigwedge_{v \in V(G)} G \models \phi(x \mapsto v). \quad (5.6)$$

As an example the complete graph satisfies the following formula

$$K_n \models \forall x, y : (\neg(x = y) \Rightarrow \text{ADJ}(x, y)). \quad (5.7)$$

There are many NIP-Complete problems that can be defined in MS, including for example 3-colorability [16]. In extensions of MS, one can also consider optimization problems such as minimum vertex cover and traveling sales person [24].

(b) MS problems and complexity

Given a MS formula, there are multiple problems one can consider. We will here be interested in model-checking, property-checking and selection problems, but will also include listing, counting and optimizing for completeness. Below we define these different problems and further details can be found in [12].

- **Model-checking:** Given a sentence ϕ and a graph, decide if $G \models \phi$.
- **Property-checking:** Given a formula $\phi(\mathcal{X})$, a graph G and an assignment α , decide if $G \models \phi(\alpha(\mathcal{X}))$.
- **Selection:** Given a formula $\phi(\mathcal{X})$ and a graph G , find an assignment α such that $G \models \phi(\alpha(\mathcal{X}))$ or if there is no such assignment output \perp .
- **Listing:** Given a formula $\phi(\mathcal{X})$ and a graph G , find the set of assignments $\{\alpha : G \models \phi(\alpha(\mathcal{X}))\}$.
- **Counting:** Given a formula $\phi(\mathcal{X})$ and a graph G , find the size of the set of assignments $\{\alpha : G \models \phi(\alpha(\mathcal{X}))\}$.
- **Optimizing:** Given a formula $\phi(\mathcal{X})$ and a graph G , find the maximum cardinality of a set assigned to X , i.e. $\max(\{|\alpha(X)| : G \models \phi(\alpha(\mathcal{X}))\})$.

In the above definitions we have only considered the problem of finding assignments to all the free variables of a formula for simplicity. One can also consider similar problems as above, where one is given a formula $\phi(\mathcal{X}, \mathcal{Y})$, a graph G , an assignment α_y to the free variables \mathcal{Y} and where the task is to find an assignment to the rest of the free variables, i.e. a α_x such that $G \models \phi(\alpha_x(\mathcal{X}), \alpha_y(\mathcal{Y}))$. It turns out that all the above problems are fixed-parameter tractable⁸ in

⁸Note that the output of the listing problem is possibly super-polynomial in the size of the graph. Thus for the listing problem, fixed-parameter tractable means polynomial scaling in the size of the input plus the size of the input.

the formula and the clique-width of the graph, as shown in [12]. The same is therefore true for rank-width, since rank-width is bounded if and only if clique-width is bounded [22].

Theorem 5.1. *There exists an algorithm which checks whether an assignment α satisfies a C_2MS formula $\phi(\mathcal{X})$ on G , i.e. whether $G \models \phi(\alpha(\mathcal{X}))$ or not, and has a running time*

$$\mathcal{O}(f(|\phi|, \text{rdw}(G)) \cdot |G|^3). \quad (5.8)$$

where $\text{rdw}(G)$ is the rank-width of G and f is an computable function.

Proof. Theorem 6.55 in [12] states that CMS model-checking problem can be solved in time

$$\mathcal{O}(f(\phi, \text{cwd}(G)) \cdot |G|^3), \quad (5.9)$$

where $\text{cwd}(G)$ is the clique-width of G . In section 6.4.1 of [12] it is also shown that the CMS property-checking problem can be reduced to the CMS model-checking problem and can therefore equivalently be solved in time as in equation (5.9). Furthermore, in definition 6.1 of [12] they show that the CMS model-checking problem can be solved in time

$$\mathcal{O}(f(|\phi|, \text{cwd}(G)) \cdot |G|^3), \quad (5.10)$$

since there are only finitely many sentences of size bounded by a given integer. The theorem then follows since the rank-width is bounded if and only if the clique-width is bounded, which is because

$$\text{rdw}(G) \leq \text{cwd}(G) \leq 2^{\text{rdw}(G)+1} - 1, \quad (5.11)$$

as shown by Oum in [22]. □

In many cases it is in fact the quantifier rank, see definition 5.1 below, which dominates the runtime to solve MS problems. For example in [25] it is shown by Langer, Rossmanith and Sikdar that the MS model-checking problem, i.e. if $G \models \phi$, can be solved in time $\mathcal{O}(f(\text{qr}(\phi), \text{cwd}(G)) \cdot |G|^3)$. As discussed with Langer, it is probably possible to extend this statement to also the CMS model-checking problem and therefore CMS property-checking problem. We therefore make the following conjecture,

Conjecture 5.1. *There exists an algorithm which checks whether an assignment α satisfies a C_2MS formula $\phi(\mathcal{X})$ on G , i.e. whether $G \models \phi(\alpha(\mathcal{X}))$ or not, and has a running time*

$$\mathcal{O}(f(\text{qr}(\phi), \text{rdw}(G)) \cdot |G|^3). \quad (5.12)$$

where $\text{qr}(\phi)$ is the quantifier rank of ϕ , $\text{rdw}(G)$ is the rank-width of G and f is a computable function.

Definition 5.1. *The quantifier rank $\text{qr}(\phi)$ of a formula ϕ is the maximum number of nested quantifiers as defined in [24] and can be found by the following recursive relations.*

$$\text{qr}(\phi) = 0, \text{ if } \phi \text{ is atomic} \qquad \text{qr}(\exists x : \phi(x)) = \text{qr}(\phi) + 1 \quad (5.13)$$

$$\text{qr}(\neg\phi) = \text{qr}(\phi) \qquad \text{qr}(\exists X : \phi(X)) = \text{qr}(\phi) + 1 \quad (5.14)$$

$$\text{qr}(\phi \vee \psi) = \max(\{\text{qr}(\phi), \text{qr}(\psi)\}) \qquad \text{qr}(\forall x : \phi(x)) = \text{qr}(\phi) + 1 \quad (5.15)$$

$$\text{qr}(\phi \wedge \psi) = \max(\{\text{qr}(\phi), \text{qr}(\psi)\}) \qquad \text{qr}(\forall X : \phi(X)) = \text{qr}(\phi) + 1 \quad (5.16)$$

(c) Vertex-minor as C_2MS formula

In [13] Courcelle and Oum show how one can express whether a graph G' is a vertex-minor of G in counting monadic second-order logic C_2MS . We restate this here and also provide the explicit C_2MS formula and its quantifier rank. In appendix B we explicitly provide all subformulas we make use of here, which can also be found in [13]. These formulas and the statement of this section build heavily on the concept of an *isotropic system* which was introduced by Bouchet in [6]. We

will not go into the details of isotropic systems here; more details can be found in [6]. The reason for the relation to isotropic systems is that an isotropic system describes an equivalence class of graphs under local complementation. Importantly, an isotropic system⁹ $S(G)$ given by a graph G has a number of *Eulerian vectors* and each of these Eulerian vectors describes a LC-equivalent graph to G . Furthermore, any LC-equivalent graph to G is described by some Eulerian vector of $S(G)$. We will here describe a Eulerian vector by three pairwise disjoint subsets of the vertices of G whose union is $V(G)$ and write this as a tuple (X_e, Y_e, Z_e) . The set of Eulerian vectors of $S(G)$ will be denoted as $\mathcal{E}(S(G))$. The formula $\text{EUL}(X_e, Y_e, Z_e)$ in equation (A 5) describes whether (X_e, Y_e, Z_e) is a Eulerian vector of $S(G)$, i.e.

$$G \models \text{EUL}(X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad (U_e, V_e, W_e) \in \mathcal{E}(S(G)). \quad (5.17)$$

As mentioned above, the set of graphs described by the Eulerian vectors of $S(G)$ are exactly the LC-equivalent graphs to G . Bouchet used this to develop an efficient algorithm to test LC-equivalence between graphs in [8], which has been used to efficiently test single-qubit Clifford equivalence of graph states in [31]. Let's denote the LC-equivalent graph to G described by the Eulerian vector (X_e, Y_e, Z_e) as $\mathcal{G}(X_e, Y_e, Z_e)$. We will now find a formula that captures whether a graph G' is a vertex-minor of G . From the above and equation (2.22) we have that $G' < G$ if and only if there exists a Eulerian vector (X_e, Y_e, Z_e) such that

$$\mathcal{G}(X_e, Y_e, Z_e)[V(G')] = G'. \quad (5.18)$$

How do we express this as a C_2MS formula? The formula $\text{ADJ}(u, v, X_e, Y_e, Z_e)$ in equation (A 7) describes whether the edge (u, v) is an edge of the graph $\mathcal{G}(X_e, Y_e, Z_e)$, i.e.

$$G \models \text{ADJ}(x \mapsto u, y \mapsto v, X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad (u, v) \in E(\mathcal{G}(U_e, V_e, W_e)). \quad (5.19)$$

Using equation (5.19) we can express whether G' is a vertex-minor of G , as described in the following theorem.

Theorem 5.2. *For any G' with vertex-set $V(G') = \{x_1, \dots, x_k\}$, there exists a C_2MS formula $\text{VM}_{G'}(x_1, \dots, x_k)$ such that*

$$G \models \text{VM}_{G'}(\alpha(x_1), \dots, \alpha(x_k)) \quad \text{iff} \quad \alpha(G') < G, \quad (5.20)$$

where x_i are the free variables of the formula, α is a bijection from $V(G')$ to a subset of $V(G)$ of size k . In equation (5.20), α functions both as an assignment of the free variables of VM and as a relabeling of the vertices in G' by $\alpha(G)$. This dual-purpose of α is valid since we identify the free variables of VM with the vertices of G' . To be precise, by $\alpha(G)$ we mean the graph

$$\alpha(G) = (\{\alpha(x) : x \in V(G')\}, \{(\alpha(x), \alpha(y)) : (x, y) \in E(G')\}). \quad (5.21)$$

Furthermore the length and quantifier rank of VM has the following scaling

$$|\text{VM}_{G'}| = \mathcal{O}(|G'|^2), \quad \text{qr}(\text{VM}_{G'}) = 10 = \mathcal{O}(1). \quad (5.22)$$

Proof. We prove this by explicitly providing the C_2MS formula as follows

$$\begin{aligned} \text{VM}_{G'}(x_1, \dots, x_k) = \exists X_e, Y_e, Z_e : \left(\text{EUL}(X_e, Y_e, Z_e) \wedge \bigwedge_{(x,y) \in E(G')} \text{ADJ}(x, y, X_e, Y_e, Z_e) \right. \\ \left. \wedge \bigwedge_{(x,y) \notin E(G')} \neg \text{ADJ}(x, y, X_e, Y_e, Z_e) \right) \end{aligned} \quad (5.23)$$

⁹There are actually multiple isotropic system $S(G, A, B)$ related to a graph, depending on the choice of supplementary vectors A and B , see [6]. Here we consider a canonical isotropic system $S(G)$ and chose the supplementary vectors to be $A = (\omega, \dots, \omega)$ and $B = (1, \dots, 1)$, where ω is a primitive element of \mathbb{F}_4

It is then clear that equation (5.20) is true, since if $G \models \text{VM}(\alpha(x_1), \dots, \alpha(x_k))$ then we know that there exist an LC-equivalent graph to G which induced subgraph on $V(G')$ has the edge-set $\{(\alpha(x), \alpha(y)) : (x, y) \in E(G')\}$. This is precisely $\alpha(G')$ and therefore $\alpha(G') < G$. Furthermore, if $\alpha(G') < G$ then we know that there exist a Eulerian vector (U_e, V_e, W_e) such that

$$\bigwedge_{(x,y) \in E(G')} \text{ADJ}(\alpha(x), \alpha(y), U_e, V_e, W_e) \wedge \bigwedge_{(x,y) \notin E(G')} \neg \text{ADJ}(\alpha(x), \alpha(y), U_e, V_e, W_e) \quad (5.24)$$

is true.

Next we show how the length and quantifier rank of VM scale with G' . Firstly the length clearly scales as

$$|\text{VM}| = \mathcal{O}(|E(G')|) = \mathcal{O}(|G'|^2). \quad (5.25)$$

The quantifier ranks of the subformulas used here are given in appendix B and in particular we have that $\text{qr}(\text{EUL}) = 7$ and $\text{qr}(\text{ADJ}) = 7$. Thus, we have that the quantifier rank of VM is

$$\text{qr}(\text{VM}) = 3 + \max(\{\text{qr}(\text{EUL}), \text{qr}(\text{ADJ})\}) = 3 + \max(\{7, 7\}) = 10 = \mathcal{O}(1). \quad (5.26)$$

□

It is easy to see that theorem 1.1 is a direct consequence of theorem 2.2, theorem 5.1 and theorem 5.2.

As mentioned earlier, it is possible to specify in C_2MS whether a graph has a vertex-minor on a subset of the vertices with a given property. We capture this in the following theorem.

Theorem 5.3. *Given a C_2MS sentence ϕ specifying some graph property P , then there exists a C_2MS formula $\text{PROP_VM}_\phi(X)$ capturing whether a graph has a vertex-minor on X which satisfies P , i.e.*

$$G \models \text{PROP_VM}_\phi(X \mapsto U) \quad \text{iff} \quad \exists G' : (V(G') = U) \wedge (G' < G) \wedge (G' \models \phi) \quad (5.27)$$

Proof. Let ϕ' be the formula made from ϕ by replacing all instances of the predicate $\text{ADJ}(x, y)$ by the formula $\text{ADJ}(x, y, X_e, Y_e, Z_e)$. If (U_e, V_e, W_e) is a Eulerian vector of $S(G)$, then we have that

$$G \models \phi'(X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad \mathcal{G}(U_e, V_e, W_e) \models \phi. \quad (5.28)$$

The expression on the right of the above equation states that the LC-equivalent graph $\mathcal{G}(U_e, V_e, W_e)$ models ϕ , but what we want is that the induced subgraph $\mathcal{G}(U_e, V_e, W_e)[X]$ models ϕ . This can be done by restricting all the quantifiers in ϕ' to the set X . Thus, let ϕ'' be the formula made from ϕ' by making the following changes to all the quantifiers

$$\forall y : \psi(y) \quad \rightarrow \quad \forall y : (y \in X \Rightarrow \psi(y)) \quad (5.29)$$

$$\forall Y : \psi(Y) \quad \rightarrow \quad \forall Y : (Y \subseteq X \Rightarrow \psi(Y)) \quad (5.30)$$

$$\exists y : \psi(y) \quad \rightarrow \quad \exists y : (y \in X \wedge \psi(y)) \quad (5.31)$$

$$\exists Y : \psi(Y) \quad \rightarrow \quad \exists Y : (Y \subseteq X \wedge \psi(Y)). \quad (5.32)$$

We then see that if (U_e, V_e, W_e) is an Eulerian vector of $S(G)$, then

$$G \models \phi''(X \mapsto U, X_e \mapsto U_e, Y_e \mapsto V_e, Z_e \mapsto W_e) \quad \text{iff} \quad \mathcal{G}(U_e, V_e, W_e)[U] \models \phi. \quad (5.33)$$

The formula PROP_VM can then be built by checking if there exists a Eulerian vector such that G models ϕ'' , i.e.

$$\text{PROP_VM}_\phi(X) = \exists X_e, Y_e, Z_e : \left[\text{EUL}(X_e, Y_e, Z_e) \wedge \phi''(X, X_e, Y_e, Z_e) \right]. \quad (5.34)$$

□

Since it is possible to specify whether a graph is a complete graph in C_2MS , see equation (5.7), we see that theorem 1.2 directly follows from theorem 2.2, theorem 5.1 and theorem 5.3. Let's use the method in the proof of theorem 5.3 to explicitly find the formula expressing whether a

graph has the complete graph as a vertex-minor on the subset X . Let COMPLETE be the formula in equation (5.7) which is modeled by G if G is a complete graph. We first find COMPLETE' by replacing the predicate ADJ(x, y)

$$\text{COMPLETE}'(X_e, Y_e, Z_e) = \forall x, y : (\neg(x = y) \Rightarrow \text{ADJ}(x, y, X_e, Y_e, Z_e)). \quad (5.35)$$

Then replacing the quantifiers as in equations (5.29)-(5.32) we get the formula

$$\begin{aligned} \text{COMPLETE}''(X, X_e, Y_e, Z_e) &= \forall x : \left[x \in X \Rightarrow \left(\forall y : y \in X \Rightarrow (\neg(x = y) \Rightarrow \text{ADJ}(x, y, X_e, Y_e, Z_e)) \right) \right] \\ &= \forall x, y : \left[((x \in X) \wedge (y \in X) \wedge \neg(x = y)) \Rightarrow \text{ADJ}(x, y, X_e, Y_e, Z_e) \right] \end{aligned} \quad (5.36)$$

Finally by using the equation (5.34) we arrive at the formula

$$\text{COMPLETE_VM}(X) = \exists X_e, Y_e, Z_e : \left[\text{EUL}(X_e, Y_e, Z_e) \wedge \text{COMPLETE}''(X, X_e, Y_e, Z_e) \right] \quad (5.37)$$

which has the following property

$$G \models \text{COMPLETE_VM}(X \mapsto U) \quad \text{iff} \quad K_U < G \quad (5.38)$$

where K_U is the complete graph with vertex-set U .

(d) Finding the sequence of operations

In this section so far, we have looked at the problem of deciding whether G' is a vertex-minor of G , but if this is true then how does one find the sequence of operations that takes G to G' ? Similarly, if $|G'\rangle$ is a qubit-minor of $|G\rangle$, what sequence of operations takes $|G\rangle$ to $|G'\rangle$? We will here describe two ways to find the sequence of operations.

Method 1: The first way is slightly simpler but increases the runtime from the decision problem by a factor of $|G|$. The idea is to use an algorithm that solves the decision problem of whether G' is a vertex-minor of G to iteratively find the sequence of operations. Let's therefore assume that we know that $G' < G$. Furthermore, let v be a vertex in $V(G) \setminus V(G')$. From theorem 3.1 we know that G' is a vertex-minor of at least one of the three graphs $G \setminus v$, $\tau_v(G) \setminus v$ or $T_v(G) \setminus v$. By using an algorithm for the decision problem we can decide which of these three graphs has G' as a vertex-minor. Let's denote one of these graphs by G_1 and the operation that takes G to G_1 as P_1 , i.e. $G_1 = P_1(G)$. That is, P_1 is either $(_) \setminus v$, $\tau_v(_) \setminus v$ or $T_v(_) \setminus v$, such that $G' < G_1 < G$. Now perform the same step again to find an operation P_2 taking G_1 to a graph G_2 which has G' as a vertex-minor. Perform the step $n - k$ times, where $n = |G|$ and $k = |G'|$. It is then clear that the sequence $P = P_{n-k} \circ \dots \circ P_1$ takes G to a LC-equivalent graph of G' . From P it is easy to find the induced sequence of local complementations m such that $\tau_m(G)[V(G')] \sim_{\text{LC}} G'$. Finally we can use the algorithm in [8] to find a sequence of local complementations m' such that $\tau_{m'} \circ \tau_m(G)[V(G')] = G'$. Assume that $|G'|$ and $\text{rwd}(G)$ are bounded¹⁰. Then according to theorem 1.1 the decision problem can be solved in time $\mathcal{O}(n^3)$. To find P we need to run the algorithm for the decision problem $\mathcal{O}(n - k)$ times and compute $P_i(G_{i-1})$ the same number of times. Computing $P_i(G_{i-1})$ can be done in time $\mathcal{O}(n^2)$, as described in section 23. Lastly finding the sequence m' can be done in time $\mathcal{O}(k^4)$. Thus, the total runtime is

$$\mathcal{O}((n^3 + n^2)(n - k)) + \mathcal{O}(k^4) = \mathcal{O}(n^4) \quad (5.39)$$

Method 2: Another way to find the sequence of operations that takes G to G' , given that $G' < G$, is to formulate the problem as a C_2 MS selection problem as described in section 5(b). Recall that an algorithm that solves the C_2 MS selection problem takes as input a formula ϕ with free variables \mathcal{X}, \mathcal{Y} , an assignment α_x to the free variables \mathcal{X} and a graph G and returns an assignment α_y such that $G \models \phi(\alpha_x(\mathcal{X}), \alpha_y(\mathcal{Y}))$ or returns \perp if no such assignment exists. We will now formulate a selection problem by making X_e, Y_e and Z_e free variables in equation (5.20) instead of quantifier

¹⁰If conjecture 5.1 is true, then $|G'|$ does not need to be bounded.

variables. Therefore, let H be an isomorphic graph to G' with vertex-set $V(H) = \{x_1, \dots, x_k\}$ and define the C_2MS formula

$$VM'_H(x_1, \dots, x_k, X_e, Y_e, Z_e) = \text{EUL}(X_e, Y_e, Z_e) \wedge \bigwedge_{(x,y) \in E(H)} \text{ADJ}(x, y, X_e, Y_e, Z_e) \quad (5.40)$$

Let α_x be a bijection from $V(H)$ to $V(G')$ such that $\alpha_x(H) = G'$. We then have the following property of the formula VM'

$$\exists \alpha_e : G \models VM'_H(\alpha_x(x_1, \dots, x_k), \alpha_e(X_e, Y_e, Z_e)) \quad \text{iff} \quad G' < G. \quad (5.41)$$

From theorem 6.55 and similar arguments as in the proof of theorem 5.1 we know that we can solve the selection problem in time $\mathcal{O}(n^3)$ if k and $\text{rwd}(G)$ are bounded, where $n = |G|$ and $k = |G'|$. Thus, given an assignment to the free variables (X_e, Y_e, Z_e) , i.e. given an Eulerian vector (U_e, V_e, W_e) such that $\mathcal{G}(U_e, V_e, W_e)[V(G')] = G'$, the question is then how to find a sequence of local complementations that takes G to $\mathcal{G}(U_e, V_e, W_e)$. In appendix C we show how to do this in time $\mathcal{O}(n)$, which shows that the total runtime is

$$\mathcal{O}(n^3) + \mathcal{O}(n) = \mathcal{O}(n^3). \quad (5.42)$$

6. Discussion

The problem of deciding whether a stabilizer state $|S_t\rangle$ can be obtained from another $|S_s\rangle$ by single-qubit Clifford operations, single-qubit Pauli measurement and classical communication is equivalent to deciding if some graph G' is a vertex-minor of another graph G . We showed here that the vertex-minor problem can be solved in cubic time in the size of G on instances where G has bounded rank-width and G' has bounded size, by using the theory of monadic second-order logic and a version of Courcelle's theorem. Furthermore, if conjecture 5.1 is true then the vertex-minor problem can be solved in cubic time in the size of G on the strictly larger class of instances where G has bounded rank-width and G' is arbitrary. A direct implementation of Courcelle's theorem is however not practical, due to a huge constant factor in the runtime of the algorithm. Finding more tailored algorithms for the vertex-minor problem on graphs of bounded rank-width is therefore of value. In [14] we provide an efficient algorithm for graphs of rank-width one, which does not have a huge constant factor in the runtime.

Given some graph property P expressible in C_2MS one can decide if a graph G has a vertex-minor on a subset $U \subseteq V(G)$ that satisfies P , in cubic time in the size of G for graphs with bounded rank-width, as we show in section 5(c). The graph property P can be for example that the graph is a complete graph or that it is k -colourable. Testing for for example 2-colourable qubit-minors could be interesting in the context of purification since it has been shown that the purification schemes in [4] purify all 2-colourable graph states.

In section 5(d) we also showed how to find the sequence of operations that take $|G\rangle$ to its qubit-minor $|G'\rangle$. Finally in section 4 we showed how these operations can be applied in constant time in the size of the graph states and how this can be done without destroying all the rest of the entanglement in the source state. An open question is how to find an optimal sequence of operations that destroys a minimum amount of entanglement in the rest of the state.

Data Accessibility. The implementation of concepts and algorithms, written in SAGE and MONA, can be freely accessed from the git-repository at [1].

Authors' Contributions. Axel Dahlberg developed the theory and proofs, drafted most of the manuscript and wrote the code at [1]. Stephanie Wehner defined and supervised the project. Both authors read and approved the manuscript.

Funding. Axel Dahlberg and Stephanie Wehner were supported by STW Netherlands, and NWO VIDI grant, and an ERC Starting grant.

Acknowledgements. We thank Alexander Langer, Robert Ganian, Kenneth Goodenough, Jonas Helsen, Tim Coopmans, Jéri my Ribiero and Ben Criger for interesting discussions and feedback.

A. Corrections from sequence of Pauli Z measurement

By performing a measurement in the standard basis of a qubit v which is part of a graph state $|G\rangle$, one can effectively disconnect qubit v from the rest of the state and produce the state $|0\rangle_v \otimes |G \setminus v\rangle$. Depending on the measurement outcome, certain single-qubit Clifford operations need to be performed to map the post-measurement state to $|0\rangle_v \otimes |G \setminus v\rangle$, as described in section 2(d). One can therefore effectively cut out a graph state on a subset of the qubits V' , i.e. producing the state $|G[V']\rangle$, by measuring the qubits in $V(G) \setminus V'$ in the standard basis and performing certain single-qubit Clifford operations. Here we show what corrections need to be applied to the qubits V' , such that the post-measurement state is mapped to the state $|G[V']\rangle$.

Let's assume that $|G\rangle$ is a graph state and we wish to transform this to $|G[V']\rangle$, by measuring the qubits $U = V(G) \setminus V'$ in the standard basis. Let's denote the qubits in U as $\{v_1, v_2, \dots, v_{n-k}\}$ and the measurement outcome of qubit v_i by¹¹ $x_i \in \{0, 1\}$. Furthermore, let's denote the projectors in the Z basis as $P_v^{(0)} = P_v^{(Z,+)}$ and $P_v^{(1)} = P_v^{(Z,-)}$. The post-measurement state is then given by

$$|\psi_{\text{post}}^{n-k}\rangle = 2^{n-k} P_{v_{n-k}}^{(x_{n-k})} \dots P_{v_2}^{(x_2)} P_{v_1}^{(x_1)} |G\rangle. \quad (\text{A } 1)$$

By acting with the projectors on $|G\rangle$ we find by induction on $n - k$ that the post-measurement state can be evaluated to

$$|\psi_{\text{post}}^{n-k}\rangle = \left(\bigotimes_{i=1}^{n-k} Z^{\sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) \otimes \left(\left(\prod_{i=1}^{n-k} (Z[N_{v_i} \cap V'])^{x_i} \right) |G[V']\rangle \right), \quad (\text{A } 2)$$

where $\text{adj}(u, v)$ is 1 if (u, v) is an edge in G and zero otherwise and where $Z[X]$ is $\prod_{x \in X} Z_x$. One can see this by checking that indeed

$$2P_{v_{n-k}}^{(x_{n-k})} |\psi_{\text{post}}^{n-k-1}\rangle = |\psi_{\text{post}}^{n-k}\rangle. \quad (\text{A } 3)$$

by using equations (2.15) and (2.19). The operations on the qubits in $V(G) \setminus V'$ in the left part of equation A 2 will only give a global phase as follows

$$\begin{aligned} \left(\bigotimes_{i=1}^{n-k} Z^{\sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) &= \left(\bigotimes_{i=1}^{n-k} (-1)^{x_i \sum_{j=1}^{i-1} x_j \text{adj}(v_i, v_j)} |x_i\rangle_{v_i} \right) \\ &= (-1)^{\sum_{i=1}^{n-k} \sum_{j=1}^{i-1} x_i x_j \text{adj}(v_i, v_j)} \bigotimes_{i=1}^{n-k} |x_i\rangle_{v_i}. \end{aligned} \quad (\text{A } 4)$$

The exponent in the global phase $\sum_{i=1}^{n-k} \sum_{j=1}^{i-1} x_i x_j \text{adj}(v_i, v_j)$ is in fact the number of edges in the induced graph $G_x = G[\{v_i : x_i = 1\}]$. Let's now consider the correction operators in the right part of equation A 2. A qubit $v \in V'$ will have a Z contribution from the i th factor if $v \in N_{v_i}$ and $x_i = 1$. Thus, we see that the total contribution on qubit v is given by

$$Z^{y_v} \quad \text{where} \quad y_v = \sum_{i \in \{i : v_i \in N_v \setminus V'\}} x_i \quad (\text{A } 5)$$

Finally we find that the post-measurement state from equation A 2 is given by

$$|\psi_{\text{post}}^{n-k}\rangle = (-1)^{|E(G_x)|} \left(\bigotimes_{i=1}^{n-k} |x_i\rangle_{v_i} \right) \otimes \left(\left(\prod_{v \in V'} Z_v^{y_v} \right) |G[V']\rangle \right). \quad (\text{A } 6)$$

¹¹We identify 0 and 1 with the measurement outcomes +1 and -1, respectively.

B. Vertex-minor formula

Here we provide the C_2MS formulas¹² which we make use of in section 5(c). We state what the formula expresses and its quantifier rank in table 1.

Table 1. The C_2MS formulas used in section 5(c), what they express and their quantifier rank. b_v is the unique vector in $S(G)$ with respect to (X_e, Y_e, Z_e) as defined in [13].

| Formula | True if and only if | qr |
|----------------------------------|---|----|
| DISJOINT(X, Y, Z) | X, Y and Z are pairwise disjoint | 1 |
| PART(X, Y, Z) | (X, Y, Z) is a tripartition | 1 |
| EVENINTER(Q, v) | $ N_v \cap Q = 0 \pmod{2}$ | 2 |
| MEMBER(X, Y, Z) | (X, Y, Z) is a vector of $S(G)$ | 4 |
| EUL(X_e, Y_e, Z_e) | (X_e, Y_e, Z_e) is a Eulerian vector of $S(G)$ | 7 |
| BASE(X, Y, Z, X_e, Y_e, Z_e) | (X, Y, Z) is b_v wrt. (X_e, Y_e, Z_e) in $S(G)$ | 4 |
| ADJ(u, v, X_e, Y_e, Z_e) | (u, v) is an edge of $\mathcal{G}(X_e, Y_e, Z_e)$ | 7 |

$$\text{DISJOINT}(X, Y, Z) = \forall x : (\neg(x \in X \wedge x \in Y) \wedge \neg(x \in X \wedge x \in Z) \wedge \neg(x \in Y \wedge x \in Z)) \quad (\text{A } 1)$$

$$\text{PART}(X, Y, Z) = \left(\underbrace{\forall x : (x \in X \vee x \in Y \vee x \in Z)}_{"V=X \cup Y \cup Z"} \right) \wedge \text{DISJOINT}(X, Y, Z) \quad (\text{A } 2)$$

$$\text{EVENINTER}(Q, v) = \forall R : \left(\underbrace{\forall u : u \in R \Leftrightarrow (\text{ADJ}(u, v) \wedge u \in Q)}_{"R=N_v \cap Q"} \right) \Rightarrow \text{EVEN}(R) \quad (\text{A } 3)$$

$$\begin{aligned} \text{MEMBER}(X, Y, Z) = \text{DISJOINT}(X, Y, Z) \wedge \left[\exists Q : \left(\forall v : \left((v \in X \Leftrightarrow (v \notin Q \wedge \neg \text{EVENINTER}(Q, v))) \wedge \right. \right. \right. \\ \left. \left. \left. (v \in Y \Leftrightarrow (v \in Q \wedge \text{EVENINTER}(Q, v))) \wedge \right. \right. \right. \\ \left. \left. \left. (v \in Z \Leftrightarrow (v \in Q \wedge \neg \text{EVENINTER}(Q, v))) \wedge \right. \right. \right. \\ \left. \left. \left. \left(\underbrace{\neg(v \in X \vee v \in Y \vee v \in Z)}_{"v \in V(G) \setminus (X \cup Y \cup Z)"} \Leftrightarrow (v \notin Q \wedge \text{EVENINTER}(Q, v)) \right) \right) \right] \quad (\text{A } 4) \end{aligned}$$

$$\begin{aligned} \text{EUL}(X_e, Y_e, Z_e) = & (\text{PART}(X_e, Y_e, Z_e)) \wedge \\ & \left[\forall X, Y, Z : (X \subseteq X_e \wedge Y \subseteq Y_e \wedge Z \subseteq Z_e \wedge \text{MEMBER}(X, Y, Z)) \right. \\ & \left. \Rightarrow \left(\underbrace{\forall v : \neg(v \in X \vee v \in Y \vee v \in Z)}_{"X=Y=Z=\emptyset"} \right) \right] \quad (\text{A } 5) \end{aligned}$$

¹² Note that there seems to be a typo in [13] since they use the formula $V = X_e \cup Y_e \cup Z_e$ to express that the vector (X_e, Y_e, Z_e) is complete, i.e. that each element of the vector is non-zero. This is however not true, consider for example the sets $X_e = Y_e = Z_e = V$ which corresponds to the zero-vector since $1 + \omega + \omega^2 = 0$ and is therefore not complete. Their formula for whether (X_e, Y_e, Z_e) is a Eulerian vector is on the other hand still correct since the second part of the formula can never be true for a non-complete vector (X_e, Y_e, Z_e) for which $V = X_e \cup Y_e \cup Z_e$.

$$\begin{aligned} \text{BASE}(X, Y, Z, X_e, Y_e, Z_e, v) = & \text{MEMBER}(X, Y, Z) \wedge \overbrace{(v \in X \vee v \in Y \vee v \in Z)}^{\text{"}v \in X \cup Y \cup Z\text{"}} \wedge \\ & \left[\forall u : \neg(v = u) \Rightarrow \right. \\ & \left. \left((u \in X \Rightarrow u \in X_e) \wedge (u \in Y \Rightarrow u \in Y_e) \wedge (u \in Z \Rightarrow u \in Z_e) \right) \right] \end{aligned} \quad (\text{A } 6)$$

$$\begin{aligned} \text{ADJ}(u, v, X_e, Y_e, Z_e) = & \neg(u = v) \wedge \\ & \left[\exists X, Y, Z : \left(\text{BASE}(X, Y, Z, X_e, Y_e, Z_e, v) \wedge \overbrace{(u \in X \vee u \in Y \vee u \in Z)}^{\text{"}u \in X \cup Y \cup Z\text{"}} \right) \right] \end{aligned} \quad (\text{A } 7)$$

C. Local complementations from Eulerian vector

Let's assume G is a graph, $S(G)$ its canonical isotropic system and (U_e, V_e, W_e) an Eulerian vector describing the graph $\mathcal{G}(U_e, V_e, W_e)$. We here consider the question of how to find a sequence of local complementations m , such that $\tau_m(G) = \mathcal{G}(U_e, V_e, W_e)$. In this section we will represent Eulerian vectors as vectors in \mathbb{F}_4^n instead of tripartitions of $V(G)$, where $n = |G|$. A tripartition (U_e, V_e, W_e) induces the vector

$$A(v) = \begin{cases} 1 & \text{if } v \in U_e \\ \omega & \text{if } v \in V_e \\ \omega^2 & \text{if } v \in W_e \end{cases} \quad (\text{A } 1)$$

where $A(v)$ is element v of the vector $A \in \mathbb{F}_4^n$ and ω is a primitive element of \mathbb{F}_4 . In [9] it is shown that for any Eulerian vector A of an isotropic system there exists exactly one other Eulerian vector A' which differ from A in only the element v . This other Eulerian vector A' is denoted $A * v$ and is called a switching of A . Furthermore, the switching induces a local complementation on the graphs the Eulerian vectors describe. More precisely, if the Eulerian vector A describe the graph G , then the Eulerian vector $A * v$ describe the graph $\tau_v(G)$. Thus, if we find a sequence of switchings taking the Eulerian vector describing G to the Eulerian vector describing $\mathcal{G}(U_e, V_e, W_e)$, we have also directly found a sequence of local complementations taking G to $\mathcal{G}(U_e, V_e, W_e)$. The Eulerian vector of $S(G)$ describing G is given as $A_0 = (\omega, \dots, \omega)$ and the one describing $\mathcal{G}(U_e, V_e, W_e)$ is given as in equation (A 1). A sequence of switchings taking A_0 to A can be found in linear time similarly to the method described in section 4 of [8]. The idea is to go over the vectors A_0 and A element by element and make these equal one by one. Let's consider a vertex $v \in V(G)$ and the four vectors $A_0, A_0 * v, A$ and $A * v$. These four vectors cannot all differ in the element v , since there are only three non-zero elements of \mathbb{F}_4 , i.e. $\{1, \omega, \omega^2\}$. Repeating this process for all elements of $V(G)$ will give two sequences of switchings, m_1 and m_2 , one for A_0 and one for A such that

$$A_0 * m_1 = A * m_2. \quad (\text{A } 2)$$

Since the switchings are involutions we have that

$$A_0 * (m_1 \bar{m}_2) = A \quad (\text{A } 3)$$

and therefore that

$$\tau_{m_1 \bar{m}_2}(G) = \mathcal{G}(U_e, V_e, W_e), \quad (\text{A } 4)$$

where $m_1 \bar{m}_2$ is the sequence m_1 followed by the reversal of m_2 . Finding $m = m_1 \bar{m}_2$ thus takes time $\mathcal{O}(n)$.

References

1. Git-repository with implemented code.
<https://github.com/AckslD/vertex-minors>.
2. MONA.
<http://www.brics.dk/mona/index.html>.
3. SAGE.
<http://www.sagemath.org/>.
4. H. Aschauer, W. Dür, and H. J. Briegel.
Multiparticle entanglement purification for two-colorable graph states.
Physical Review A - Atomic, Molecular, and Optical Physics, 71(1):1–22, 2005.
5. K. Azuma, K. Tamaki, and H. K. Lo.
All-photonic quantum repeaters.
Nature Communications, 6:1–7, 2015.
6. A. Bouchet.
Isotropic Systems.
European Journal of Combinatorics, 8(3):231–244, 1987.
7. A. Bouchet.
Graphic presentations of isotropic systems.
Journal of Combinatorial Theory, Series B, 45(1):58–76, 1988.
8. A. Bouchet.
An efficient algorithm to recognize locally equivalent graphs.
Combinatorica, 11(4):315–329, dec 1991.
9. A. Bouchet.
Recognizing locally equivalent graphs.
Discrete Mathematics, 114(1-3):75–86, apr 1993.
10. M. Christandl and S. Wehner.
Quantum Anonymous Transmissions.
In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3788 LNCS, pages 217–235. 2005.
11. B. Courcelle.
The Monadic Second-Order Theory of Graphs. I. Recognizable Sets of Finite graphs.
Information and Computation, 85(1):12–75, 1990.
12. 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.
13. 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.
14. A. Dahlberg, J. Helsen, and S. Wehner.
How to transform graph state using single-qubit operations: computational complexity and algorithms.
arXiv pre-print: 1805.05306, 2018.
15. R. G. Downey and M. R. Fellows.
Parameterized Complexity.
Proc 6th Annu Conf on Comput Learning Theory, 5(1):51–57, 1999.
16. 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.
17. D. Gottesman.
Stabilizer Codes and Quantum Error Correction.
PhD thesis, may 1997.
18. D. Gottesman.
The Heisenberg Representation of Quantum Computers.
Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, 1:32–43, 1999.
19. A. Gupta and N. Nishimura.
The complexity of subgraph isomorphism for classes of partial k-trees.
Theoretical Computer Science, 164(1-2):287–298, 1996.

20. M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, and H. J. Briegel.
Entanglement in Graph States and its Applications.
Quantum Computers, Algorithms and Chaos, pages 1–99, 2006.
21. M. Hein, J. Eisert, and H. J. Briegel.
Multiparty entanglement in graph states.
Physical Review A - Atomic, Molecular, and Optical Physics, 69(6):062311–1, 2004.
22. S. il Oum and P. Seymour.
Approximating clique-width and branch-width.
Journal of Combinatorial Theory. Series B, 96(4):514–528, 2006.
23. R. Jozsa, D. S. Abrams, J. P. Dowling, and C. P. Williams.
Quantum Clock Synchronization Based on Shared Prior Entanglement.
Physical Review Letters, 85(9):2010–2013, aug 2000.
24. 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, nov 2014.
25. A. Langer, P. Rossmanith, and S. Sikdar.
Linear-time algorithms for graphs of bounded rankwidth: A fresh look using game theory (extended abstract).
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6648 LNCS:505–516, 2011.
26. D. Markham and B. C. Sanders.
Graph states for quantum secret sharing.
Physical Review A - Atomic, Molecular, and Optical Physics, 78(4), 2008.
27. S. I. Oum.
Rank-width and vertex-minors.
Journal of Combinatorial Theory. Series B, 95(1):79–100, 2005.
28. R. Raussendorf and H. J. Briegel.
A One-Way Quantum Computer.
Physical Review Letters, 86(22):5188–5191, may 2001.
29. J. Ribeiro, G. Murta, and S. Wehner.
Fully device-independent conference key agreement.
Physical Review A, 97(2):022307, feb 2018.
30. N. Robertson and P. Seymour.
Graph minors: Algorithmic aspects of tree-width.
J Algorithms, 7:309–322, 1986.
31. 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, sep 2004.
32. 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, feb 2004.
33. 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, jan 2007.