Span-program-based quantum algorithm for tree detection

Guoming Wang

Computer Science Division, University of California, Berkeley, U.S.A. *

Abstract

Span program is a linear-algebraic model of computation originally proposed for studying the complexity theory. Recently, it has become a useful tool for designing quantum algorithms. In this paper, we present a time-efficient span-program-based quantum algorithm for the following problem. Let $T$ be an arbitrary tree. Given query access to the adjacency matrix of a graph $G$ with $n$ vertices, we need to determine whether $G$ contains $T$ as a subgraph, or $G$ does not contain $T$ as a minor, under the promise that one of these cases holds. We call this problem the subgraph/not-a-minor problem for $T$. We show that this problem can be solved by a bounded-error quantum algorithm with $O(n)$ query complexity and $\tilde{O}(n)$ time complexity. The query complexity is optimal, and the time complexity is tight up to polylog factors.

1 Introduction

Given two graphs $G$ and $P$, it is natural to ask whether $G$ contains $P$ as a subgraph. This problem, known as the subgraph isomorphism problem, has numerous applications in cheminformatics [3], circuit design [19] and software engineering [11]. If $G$ and $P$ are both given as input, this problem is NP-complete, and hence it is unlikely to be solvable in polynomial time. However, if $P$ is fixed and only $G$ is given as input, then this problem, usually called the P-containment problem, can be solved efficiently. Specifically, if $P$ contains $k$ vertices, then the P-containment problem can be solved in $O(n^k)$ classical time, where $n$ is the number of vertices in $G$. In fact, by exploiting $P$’s structure cleverly, we can usually do much better. For example, if $P$ is a tree, then the P-containment problem can be solved in $O(n^2)$ classical time [11] (assuming $G$ is given by the $n \times n$ adjacency matrix).

Recently, there has been rising interest in developing fast quantum algorithms for the subgraph containment problem. In particular, the problem of triangle finding has received the most attention, perhaps due to its simplicity and its application to boolean matrix multiplication. Magniez, Santha and Szegedy [17] first gave two quantum algorithms for this problem, one based on Grover’s search and the other based on quantum walks [2]. They achieved $\tilde{O}(n^{13/10})$ quantum query complexity for this problem. Then, Belovs [5] used learning graphs to show that this problem has $O(n^{35/27})$ quantum query complexity. Subsequently, this result was improved to $O(n^{9/7})$, first by Lee, Magniez and Santha [16] who used learning graphs, second by Jeffery, Kothari and Magniez who used nested quantum walks [12].

There has also been work on quantum algorithms for detecting other patterns. Childs and Kothari [9] studied the quantum query complexity of detecting paths, claws, cycles and bipartite patterns, etc. Later, Belovs and Reichardt [8] showed that detecting paths and subdivided stars can be done in $O(n)$ queries and $\tilde{O}(n)$ time. Finally, there were also upper bounds on the quantum query complexity of detecting arbitrary patterns [15][26][12]. It is worth mentioning that most of the above algorithms are query-efficient but not time-efficient.

*Email: wgmcreate@berkeley.edu.
In this paper, we present a time-efficient quantum algorithm for the following variant of tree containment problem. Let $T = (V_T, E_T)$ be an arbitrary tree. Given query access to the adjacency matrix of a graph $G$ with $n$ vertices, we need to determine whether $G$ contains $T$ as a subgraph, or $G$ does not contain $T$ as a minor, under the promise that one of these cases holds. We call this problem the subgraph/not-a-minor problem for $T$. We show that this problem can be solved by a bounded-error quantum algorithm with $O(n)$ query complexity and $O(n)$ time complexity. Formally,

**Theorem 1.** Let $T = (V_T, E_T)$ be an arbitrary tree. Then the subgraph/not-a-minor problem for $T$ can be solved by a bounded-error quantum algorithm with $O(n)$ query complexity and $O(n)$ time complexity.

Note that by a reduction from the unstructured search, one can show that the subgraph/not-a-minor problem for any $T$ has $\Omega(n)$ quantum query complexity (see Proposition 4 of [8]). Thus, the query complexity of our algorithm is optimal, and its time complexity is tight up to polylog factors.

Our algorithm is developed based on span programs. Span program [14] is a linear-algebraic model of computation originally proposed for the study of complexity theory. Informally speaking, a span program is composed of a target vector and a collection of input vectors from an inner-product space. It accepts an input if and only if the target vector lies in the span of the available input vectors on this input. Its complexity is captured by a measure called witness size. In a recent breakthrough, Reichardt [20, 21] proved that for any (partial) boolean function, the optimal witness size of span programs for this function is within a constant factor of the bounded-error quantum query complexity for the same function. His result leads to a novel approach to develop quantum algorithms based on span programs. To date, several quantum algorithms [4, 8, 10, 22, 24, 23] have been designed in this way. In fact, learning graphs are a special class of span programs, and hence the learning-graph-based algorithms [7, 6, 5, 16, 15, 26] can be also viewed as span-program-based. Nevertheless, most of these algorithms are only query-efficient but not time-efficient. The efficient evaluation of span programs and learning graphs can be challenging.

Our work is closely related to the span program for undirected $st$-connectivity [8], which works as follows. In order to test whether $s$ and $t$ are connected in an undirected graph $G = (V, E)$, we build a span program with target vector $|s\rangle - |r\rangle$ and input vectors $|u\rangle - |v\rangle$ for any $u, v \in V$. The input vector $|u\rangle - |v\rangle$ is available if and only if $(u, v) \in E$. Then it is obvious that the target vector lies in the span of the available input vectors if and only if there is a path connecting $s$ and $t$ in $G$. In other words, the idea of this span program is that we try to run a single flow from $s$ to $t$ in $G$.

We utilize the span program for undirected $st$-connectivity in the task of tree detection as follows. Suppose $T$ has root $r$ and leaves $f_1, f_2, \ldots, f_k$. Then, in order to test whether $G$ contains $T$ as a subgraph, we verify that $G$ contains $k$ paths such that: (1) the $j$-th path resembles the path from $r$ to $f_j$ in $T$; (2) these $k$ paths overlap in certain way so that their union looks like $T$. To accomplish this, we use a technique called “parallel flows”. Namely, we run $k$ flows in parallel such that the $j$-th flow corresponds to the path from $r$ to $f_j$, and we let these flows interfere somehow to ensure that they respect the correlations among the $k$ paths. Algebraically, our span program is the “direct sum” of $k$ span programs for undirected $st$-connectivity, but these $k$ span-programs are also correlated somehow so that their solutions (i.e. the $st$-paths) overlap in certain way. This parallel-flow technique might be useful somewhere else (see Section 5).

For analyzing the witness size of our span program, we prove a theorem (i.e. Claim 2) about the structure of graphs that do not contain a given tree as a minor, which might be of independent interest.

---

1We use the symbol $\tilde{O}$ to suppress polylog factors. Namely, $\tilde{O}(f(n)) = O\left(f(n)(\log f(n))^b\right)$ for some constant $b \geq 0$. 

2
2 Preliminaries

2.1 Notation

Let \([n] = \{1, 2, \ldots, n\}\). For any matrix \(A\), let \(C(A)\) be the column space of \(A\), and let \(\text{Ref}_A\) be the reflection about \(C(A)\). Furthermore, let \(\text{Ker}(A)\) be the kernel of \(A\).

For any tree \(T = (V_T, E_T)\), let \(V_{T, i}\) and \(V_{T, l}\) be the set of internal nodes and leaves of \(T\) respectively. Also, let \(r\) be the root of \(T\). For any \(x \in V_T\), let \(C(x)\) be the set of \(x\)'s children, and let \(T^x\) be the subtree of \(T\) rooted at \(x\), and let \(V_T^x\) be the set of nodes in \(T^x\), and let \(V_{T, i}^x\), \(V_{T, l}^x\) be the set of internal and leaf nodes in \(T^x\) respectively. Furthermore, we will add two special nodes \(s\) and \(t\), and set \(V_{T, s}^x = V_{T, l}^y = V_{T, l}^z\). Then, for any \(x, y \in V_T \cup \{s, t\}\), let \(S_{x,y} = V_{T, s}^x \cap V_{T, l}^y\). We say \(x, y \in V_T\) \(\{s, t\}\) are adjacent (denoted by \(x \sim y\)) if \((x, y) \in E_T\) or \((x, y) = \{s, t\}\) or \((x, y) = \{s, r\}\) or \((x, y) = \{t, f\}\) for some \(f \in V_{T, f}\). Then for any \(x \in V_T \cup \{s, t\}\) and \(f \in V_{T, f}\), let \(M(x, f) = \{y \in V_T : x \sim y, f \in S_{x,y}\}\). It is easy to check that \(|M(x, f)| = 2\).

For any graph \(G = (V_G, E_G)\) and \(U \subseteq V_G\), let \(G|_U \triangleq (U, E_{G|_U})\) be the induced subgraph of \(G\) on the vertex set \(U\). Namely, for any \(u, v \in U\), \((u, v) \in E_{G|_U}\) if and only if \((u, v) \in E_G\).

2.2 Graph theory

For any two graphs \(G = (V_G, E_G)\) and \(H = (V_H, E_H)\), we say that \(H\) is a minor of \(G\) if \(H\) can be obtained from \(G\) by deleting and contracting edges of \(G\), and removing isolated vertices. Here, contracting an edge \((u, v)\) means replacing \(u\) and \(v\) by a new vertex and connecting this vertex to the original neighbors of \(u\) and \(v\).

Suppose \(E\) is an arbitrary subset of \(E_G\). Then we use \(E(G)\) to denote the graph obtained by contracting the edges in \(E\). Note that \(E(G)\) is well-defined, because the final graph is independent of the order of the edge contractions. Moreover, if the vertices \(v_1, \ldots, v_k \in V_G\) are combined together in \(E(G)\) (and no other vertex is combined with them), then we denote this new vertex as \(w \triangleq \{v_1, \ldots, v_k\}\) and we say that \(w\) contains \(v_1, \ldots, v_k\). Finally, for any \(u \in V_G\) we say that \(u\) is involved in \(E\) if there exists \(v \in V_G\) such that \((u, v) \in E\).

2.3 Span program and quantum query complexity

Span program is a linear-algebraic model of computation defined as follows:

**Definition 1** (Span program \([14]\)). A span program \(P\) is a 6-tuple \((n, d, |\tau|, \{v_j\} : j \in [m], I_{\text{free}}, \{I_{i,b} : i \in [n], b \in \{0, 1\}\}\), where \(|\tau| \in \mathbb{R}^d\), \(v_j \in \mathbb{R}^d\) for any \(j \in [m]\), and \(I_{\text{free}} \cup (\cup_{i=1}^n I_{i,x_i}) = I \triangleq [m]\). \(\tau\) is called the target vector, and each \(v_j\) is called an input vector. For any \(j \in I_{\text{free}}\), we say that \(v_j\) is a free input vector; for any \(j \in I_{i,b}\) for some \(i \in [n]\) and \(b \in \{0, 1\}\), we say that \(v_j\) is labeled by \((i, b)\).

To \(P\) corresponds a boolean function \(f_P : \{0, 1\}^n \rightarrow \{0, 1\}\) defined as follows: for \(x = x_1 \ldots x_n \in \{0, 1\}^n\),

\[
f_P(x) = \begin{cases} 
1, & \text{if } \tau \in \text{span}\left(\{v_j : j \in I_{\text{free}} \cup (\cup_{i=1}^n I_{i,x_i})\}\right), \\
0, & \text{otherwise}.
\end{cases}
\]

Namely, on input \(x\), only the \(v_j\)'s with \(j \in I_{\text{free}} \cup (\cup_{i=1}^n I_{i,x_i})\) are available, and \(f_P(x) = 1\) if and only if the target vector lies in the span of the available input vectors.

For convenience, we say that \(P\) accepts or rejects \(x\) if \(f_P(x) = 1\) or \(0\), respectively.

The complexity of a span program is measured by its witness size defined as follows:
**Definition 2 (Witness size [20]).** Let $P = (n, d, |\tau|, \{|v_j| : j \in [m]\}, I_{\text{free}}, \{I_{i,b} : i \in [n], b \in \{0, 1\}\})$ be a span program. Let $I = I_{\text{free}} \cup \bigcup_{j=1}^{n} \bigcup_{b \in \{0, 1\}} I_{i,b}$ and let $A = \sum_{j \in I} |v_j\rangle\langle j|$. Then, for any $x \in \{0, 1\}^n$, let $I(x) = I_{\text{free}} \cup \bigcup_{j=1}^{n} I_{i,x_j}$. Then, let $\Pi(x) = \sum_{j \in I(x)} |j\rangle\langle j|$, $\Pi(x) = \sum_{j \in I(x)} |j\rangle\langle j|$. The witness size of $P$ on $x$, denoted by $\text{wsize}(P,x)$, is defined as follows:

- If $f_P(x) = 1$, then $|\tau| \in C(A(\Pi(x)))$, so there exists $|w| \in \mathbb{R}^m$ satisfying $A\Pi(x)|w| = |\tau|$. Any such $|w|$ is a (positive) witness for $x$, and its size is defined as $||\Pi(x)|w||^2$. Then $\text{wsize}(P,x)$ is defined as the minimal size among all such witnesses.

- If $f_P(x) = 0$, then $|\tau| \not\in C(A(\Pi(x)))$, so there exists $|w'| \in \mathbb{R}^d$ satisfying $\langle \tau|w'\rangle = 1$ and $\Pi(x)A^+|w'\rangle = 0$. Any such $|w'|$ is a (negative) witness for $x$, and its size is defined as $||A^+|w'||^2$. Then $\text{wsize}(P,x)$ is defined as the minimal size among all such witnesses.

For any $\mathcal{D} \subseteq \{0, 1\}^n$ and $b \in \{0, 1\}$, let

$$\text{wsize}_b(P,\mathcal{D}) = \max_{x \in \mathcal{D}, f_P(x) = b} \text{wsize}(P,x). \quad (2)$$

Then the witness size of $P$ over domain $\mathcal{D}$ is defined as

$$\text{wsize}(P,\mathcal{D}) = \sqrt{\text{wsize}_0(P,\mathcal{D}) \text{wsize}_1(P,\mathcal{D})}. \quad (3)$$

Surprisingly, for any (partial) boolean function, the optimal witness size of span programs for this function is within a constant factor of the bounded-error quantum query complexity for the same function:

**Theorem 2 ([20, 21]).** For any function $f : \mathcal{D} \rightarrow \{0, 1\}$ where $\mathcal{D} \subseteq \{0, 1\}^n$, let $Q(f)$ be the bounded-error quantum query complexity of $f$. Then

$$Q(f) = \Theta\left(\inf_{P, f_P|\mathcal{D} = f} \text{wsize}(P,\mathcal{D})\right), \quad (4)$$

where the infimum is over span programs $P$ that compute a function agreeing with $f$ on $\mathcal{D}$. Moreover, this infimum is achieved.

In particular, a span program with small witness size can be converted into a quantum algorithm with small query complexity:

**Corollary 1.** For any function $f : \mathcal{D} \rightarrow \{0, 1\}$ where $\mathcal{D} \subseteq \{0, 1\}^n$, if $P$ is a span program computing a function agreeing with $f$ on $\mathcal{D}$, then there exists a bounded-error quantum algorithm that evaluates $f$ with $O(\text{wsize}(P,\mathcal{D}))$ queries.

### 3 Span program for tree detection

In this section, we build a span program for the subgraph/not-a-minor problem for any tree. This span program has witness size $O(n)$. Then it follows from Corollary 1 that this problem has $O(n)$ quantum query complexity.

**Theorem 3.** Let $T = (V_T, E_T)$ be an arbitrary tree. Then there exists a bounded-error quantum algorithm for the subgraph/not-a-minor problem for $T$ with $O(n)$ query complexity.
Proof. Let $G = (V_G, E_G)$ be a graph with $n$ vertices. We need to decide whether $G$ contains $T$ as a subgraph or $G$ does not contain $T$ as a minor, under the promise that one of the cases holds.

**Color coding.** We use the color coding technique from [1]. Namely, we map each vertex $u \in V_G$ to a uniformly random node $c(u) \in V_T$, and the vertices of $G$ are colored independently. Then, we discard all the “badly” colored edges, i.e. we remove any edge $(u, v) \in E_G$ such that $(c(u), c(v)) \notin E_T$. Let $c : V_G \to V_T$ be a random coloring, and let $G_c = (V_G, E_{G_c})$ be the colored graph corresponding to $c$.

We say that $G_c$ contains a correctly colored $T$-subgraph if if there is an injection $\iota : V_T \to V_G$ such that: (1) $c \circ \iota$ is the identity, i.e. $c(\iota(a)) = a$ for any $a \in V_T$; (2) for any $x, y \in V_T$, if $(x, y) \in E_T$, then $(\iota(x), \iota(y)) \in E_{G_c}$.

We will construct a span program that accepts if $G_c$ contains a correctly colored $T$-subgraph, and rejects if $G_c$ does not contain $T$ as a minor. Note that if $G$ contains $T$ as a subgraph, then this subgraph is colored correctly with probability at least $|V_T|^{-|V_T|} = \Omega(1)$. So $G_c$ contains $T$ as a subgraph with constant probability. On the other hand, if $G$ does not contain $T$ as a minor, then $G_c$ does not contain $T$ as a minor either. Thus, evaluating our span program for a constant number of independent colorings would suffice to detect $T$ with probability at least $2/3$.

**Span program.** Our span program $P$ is defined over the $|V_T| (n+2)$-dimensional space spanned by the vectors

$$\{|u\} \otimes |f\rangle : u \in \{s, t\} \cup V_G, f \in V_{T,l},$$

where $\langle u | v \rangle = \delta_{uv}$, for any $u, v \in \{s, t\} \cup V_G$, and $\langle f | g \rangle = \delta_{fg}$, for any $f, g \in V_{T,l}$.

The target vector of $P$ is

$$|\tau\rangle \triangleq (|s\rangle - |t\rangle) \otimes \left( \sum_{f \in V_{T,l}} |f\rangle \right).$$

The input vectors of $P$ include the following ones:

- For any $u \in c^{-1}(r)$, there is a free input vector

$$|(s, u)\rangle \triangleq (|s\rangle - |u\rangle) \otimes \left( \sum_{f \in V_{T,l}} |f\rangle \right).$$

- For any $f \in V_{T,l}$ and $u \in c^{-1}(f)$, there is a free input vector

$$|(u, t)\rangle \triangleq (|u\rangle - |t\rangle) \otimes |f\rangle.$$

- For any $x \in V_{T,i}, y \in C(x), u \in c^{-1}(x)$ and $v \in c^{-1}(y)$, there is an input vector

$$|(u, v)\rangle \triangleq (|u\rangle - |v\rangle) \otimes \left( \sum_{f \in V_{T,l}} |f\rangle \right),$$

and this input vector is available if and only if $(u, v) \in E_{G_c}$.

\[\text{Note that we can determine whether an edge is present in } G_c \text{ or not by querying the presence of this edge in } G \text{ and using the information about } c.\]
Here is a more compact way to describe these input vectors. We add two special vertices $s$ and $t$ to $G_c$, and color $s$, $t$ as themselves, i.e. $c(s) = s$ and $c(t) = t$. Furthermore, we connect $s$ to the vertices in $c^{-1}(r)$, and connect $t$ to the vertices in $c^{-1}(f)$ for any $f \in V_T$. Let $G'_c$ be this modified graph. For any $x \in V_T \cup \{s,t\}$, we call $c^{-1}(x)$ a block. Then $G'_c$ contains only edges between adjacent blocks. Namely, $c^{-1}(x)$ and $c^{-1}(y)$ are adjacent if and only if $x$ and $y$ are adjacent, i.e. $x \sim y$. Then, for any $u, v$ in adjacent blocks, we have an input vector

$$
| (u,v) \rangle = (| u \rangle - | v \rangle) \otimes \left( \sum_{f \in S(c(u),c(v))} | f \rangle \right),
$$

(10)

and this input vector is available if and only if the edge $(u,v)$ is present in $G'_c$.

**An example.** Let $T$ be the complete 2-level binary tree, and let $G$ be a 12-vertex graph shown in Fig. 1(b). Let $c : V_G \rightarrow V_T$ be a coloring defined as $c(u_1) = c(u_2) = r$, $c(u_3) = c(u_4) = d_1$, $c(u_5) = c(u_6) = d_2$, $c(u_7) = f_1$, $c(u_8) = c(u_9) = f_2$, $c(u_{10}) = c(u_{11}) = f_3$, $c(u_{12}) = f_4$. Then, after removing the badly colored edges (such as $(u_2,u_8)$), the colored graph $G_c$ is shown in Fig. 1(c).

![Figure 1: An example of coloring coding.](image)

Then the span program $\mathcal{P}$ is defined as follows:

- **Target vector:** $(| s \rangle - | t \rangle) \otimes (\sum_{j \in [4]} | f_j \rangle)$;
- **Free input vectors:**
  
  - $(| s \rangle - | u_i \rangle) \otimes (\sum_{j \in [4]} | f_j \rangle)$, for $i \in [2]$;
  
  - $(| u_i \rangle - | r \rangle) \otimes | f_{\chi(j)} \rangle$, for $j \in \{7, 8, \ldots, 12\}$, where $\chi(7) = 1$, $\chi(8) = 2$, $\chi(9) = 3$, $\chi(10) = 4$;
- **Other input vectors:**
  
  - $(| u_i \rangle - | u_j \rangle) \otimes (| f_i \rangle + | f_2 \rangle)$ for $i \in \{1, 2\}$ and $j \in \{3, 4\}$;
  
  - $(| u_i \rangle - | u_j \rangle) \otimes (| f_3 \rangle + | f_4 \rangle)$ for $i \in \{1, 2\}$ and $j \in \{5, 6\}$;
  
  - $(| u_i \rangle - | u_j \rangle) \otimes | f_1 \rangle$ for $i \in \{3, 4\}$;
  
  - $(| u_i \rangle - | u_j \rangle) \otimes | f_2 \rangle$ for $i \in \{3, 4\}$ and $j \in \{8, 9\}$;
– $(\ket{u_i} - \ket{u_j}) \otimes \ket{f_3}$ for $i \in \{5, 6\}$ and $j \in \{10, 11\}$;
– $(\ket{u_i} - \ket{u_{12}}) \otimes \ket{f_4}$ for $i \in \{5, 6\}$.

**Witness size.** Next, we will show that our span program $\mathcal{P}$ indeed solves the subgraph/not-a-minor problem for $T$, and along the way we also obtain upper bounds on the positive and negative witness sizes of $\mathcal{P}$. We will consider the positive and negative cases separately.

**Positive case.** Let us first consider the positive case, i.e. $G_c$ contains $T$ as a subgraph.

**Lemma 1.** Suppose $G_c$ contains $T$ as a subgraph. Then $\mathcal{P}$ accepts $G_c$. Moreover, the witness size of $\mathcal{P}$ on $G_c$ is $\mathcal{O}(1)$.

**Proof.** Suppose $G_c$ contains $T$ as a subgraph. Then there exists an injection $\iota : V_T \to V_G$ such that, for any $a \in V_T$, and $b \in C(a)$, $(\iota(a), \iota(b)) \in E_{G_c}$ and hence the input vector $\ket{(\iota(a), \iota(b))}$ is available. Thus, the target vector $\ket{\tau}$ can be written as

$$\ket{\tau} = \ket{(s, 1(r))} + \sum_{a \in V_T, b \in C(a)} \ket{(\iota(a), \iota(b))} + \sum_{f \in V_T} \ket{(\iota(f), t)}.$$  \hspace{1cm} (11)

So $\mathcal{P}$ accepts $G_c$. Furthermore, the witness size of $\mathcal{P}$ on $G_c$ is $|E_T| = \mathcal{O}(1)$, since $T$ has constant size.

For example, consider the $T$ and $G_c$ in Fig.1. $G_c$ contains $T$ as a subgraph, and $\mathcal{P}$ accepts $G_c$. The solution is

$$\ket{\tau} = \ket{(s, u_1)} + \ket{(u_1, u_3)} + \ket{(u_1, u_6)} + \ket{(u_3, u_7)} + \ket{(u_3, u_9)} + \ket{(u_6, u_{11})} + \ket{(u_6, u_{12})} + \ket{(u_7, t)} + \ket{(u_9, t)} + \ket{(u_{11}, t)} + \ket{(u_{12}, t)}.$$  \hspace{1cm} (12)

This solution can be graphically represented by four “parallel” flows from $s$ to $t$ shown in Fig.2.

![Figure 2](image)

Figure 2: A graphical representation of the solution to the span program $\mathcal{P}$ in the positive case. For each edge used by the flows, the coefficient for the corresponding input vector is +1. The coefficient for any other input vectors is 0.

**Negative case.** Now let us consider the negative case, i.e. $G_c$ does not contain $T$ as a minor.

But first, let us explain why we do not just consider the case $G_c$ does not contain $T$ as a subgraph. The problem is that the span program $\mathcal{P}$ fails to solve this problem in general. Namely,
**Claim 1.** There exists some $G_c$ which does not contain $T$ as a subgraph but $P$ accepts it.

**Proof.** For example, consider the $T$ and $G_c$ in Fig.3. Here $T$ is the complete 2-level binary tree, and $G_c$ is a 12-vertex graph shown in Fig.3. $G_c$ does not contain $T$ as a subgraph, but $P$ accepts $G_c$. The solution is

$$|	au| = |(s,u_1)| - |(s,u_2)| + |(s,u_3)| + |(u_1,u_5)| + |(u_1,u_6)| - |(u_2,u_6)| + |(u_2,u_7)| + |(u_3,u_8)| + |(u_5,u_9)| + |(u_5,u_{10})| + |(u_8,u_{11})| + |(u_8,u_{12})| + |(u_9,t)| + |(u_{10},t)| + |(u_{11},t)| + |(u_{12},t)|.$$  \hspace{1cm} (13)

This solution can be graphically represented as four flows from $s$ to $t$ shown in Fig.3. Notice that these flows move back and forth between adjacent “layers”, where each layer is colored by the nodes at the same depth of $T$. Also, note that if we contract the edges $(u_1,u_6), (u_2,u_6), (u_2,u_7)$ and $(u_3,u_7)$ of $G_c$, we would get a new graph isomorphic to $T$. In other words, $G_c$ contains $T$ as a minor.

![Figure 3: An example showing that the span program $P$ does not solve the tree containment problem in general. Here $G_c$ does not contain $T$ as a subgraph, but $P$ accepts it. The solution is graphically illustrated by Fig.3. For each edge used by the flows, the coefficient for the corresponding input vector is $+1$ or $-1$, depending on whether the flow moves towards $t$ or $s$, respectively. The coefficient for any other input vector is $0$.](image)

We can directly prove that if $P$ accepts $G_c$, then $G_c$ must contain $T$ as a minor. But here we prefer to use a different approach. We will assume that $G_c$ does not contain $T$ as a minor. Then, we will show that $P$ must reject $G_c$ by explicitly giving a negative witness for $G_c$. The advantage of this approach is that we not only get to know that $P$ accepts only certain graphs that contain $T$ as a minor, but also obtain an upper bound on the negative witness size of $P$.

Now let us return to the negative case, i.e. $G_c$ does not contain $T$ as a minor. For convenience, we introduce the following notation. For any $(a,b) \in E_T$ and $u \in c^{-1}(a)$, let $N_{a \rightarrow b}(u) = \{ v \in c^{-1}(b) : (u,v) \in E_{G_c}\}$. For any $U \subseteq c^{-1}(a)$, let $N_{a \rightarrow b}(U) = \cup_{u \in U} N_{a \rightarrow b}(u)$. For any $a \in V_T$, let $Y_a = c^{-1}(V_T^a) = \cup_{b \in V_T} c^{-1}(b)$ and let $G^a_c = G_c|y_a$.

We claim that $G_c$ must satisfy the following property:
Claim 2. For any graph $G = (V_G, E_G)$ and coloring $c : V_G \rightarrow V_T$, if $G_c = (V_G, E_G)$ does not contain $T$ as a minor, then there exist $\{v_a \subseteq c^{-1}(a) : a \in V_T\}$ and $\{v_{a,b} \subseteq c^{-1}(a) : a \in V_T, b \in C(a)\}$ such that

\begin{enumerate}
  \item $\forall a \in V_T$, $V_a$ is the disjoint union of $V_{a,b}$'s for $b \in C(a)$;
  \item $V_r = c^{-1}(r)$;
  \item $\forall a \in V_T$, $\forall b \in C(a) \cap V_T$, $N_{a \rightarrow b}(V_{a,b}) \subseteq V_b$ and $N_{b \rightarrow a}(V_{b,a}) \subseteq V_{a,b}$;
  \item $\forall a \in V_T$, $\forall b \in C(a) \cap V_T$, $N_{a \rightarrow b}(V_{a,b}) = \emptyset$.
\end{enumerate}

\textbf{Proof.} See Appendix A

Intuitively, the $V_a$'s contain the “bad” vertices that are responsible for the fact that $G_c$ does not contain $T$ as a minor. For any vertex $u \in V_a$, there is no subgraph of $G^u$ that can be contracted into a tree rooted at $u$ and isomorphic to $T^u$. In particular, since $G_c$ does not contain $T$ as a minor, for any vertex $u \in c^{-1}(r)$, there is no subgraph of $G_c$ that can be contracted into a tree rooted at $u$ and isomorphic to $T$, and hence $V_r = c^{-1}(r)$. Furthermore, if $u \in V_{a,b} \subseteq V_a$ for some $b \in C(a)$, then $u$ is “bad” because its “children” in $c^{-1}(b)$ are “bad”. Namely, if we attempt to use $u$ as the root of $T^b$, then we will fail because we will not be able to get a complete $T^b$ (which is a subtree of $T^c$). In the degenerate case, $b$ is a leaf, and $u \in V_{a,b}$ if and only if it has no neighbor in $c^{-1}(b)$. Condition 3 tells us that these “bad” vertices form a “connected component” in some sense, and it is crucial for bounding the witness size. Fig.4 demonstrates an example of such $V_a$'s and $V_{a,b}$'s.

![Diagram](image-url)

Figure 4: In this example, $T$ is the full 3-level binary tree, and $G_c$ does not contain $T$ as a minor. We have $V_r = \{u_1, u_2, u_3\}$, $V_{r,d_1} = \{u_3\}$, $V_{r,d_2} = \{u_1, u_2\}$, $V_{d_1} = V_{d_1,b_1} = \{u_5\}$, $V_{d_2} = \{u_6, u_7\}$, $V_{d_2,b_3} = \{u_7\}$, $V_{d_2,b_4} = \{u_6\}$, and any other $V_a$ or $V_{a,b}$ is $\emptyset$. Note that, $N_{r \rightarrow d_2}(V_{r,d_2}) = \{u_6, u_7\} = V_{d_2}$, $N_{d_2 \rightarrow b_3}(V_{d_2,b_3}) = \emptyset = V_{b_3}$ and $N_{d_2 \rightarrow b_4}(V_{d_2,b_4}) = \emptyset = V_{b_4}$, etc.

Lemma 2. Suppose $G_c$ does not contain $T$ as a minor. Then $P$ rejects it. Moreover, the witness size of $P$ on $G_c$ is $O(n^2)$.
Proof. Using Claim 2, we build a negative witness for $G_c$ as follows. Let

$$|w\rangle = |w_s\rangle + \sum_{a \in V_{T,I}} |w_a\rangle,$$

where

$$|w_s\rangle = |s\rangle \otimes \left( \frac{1}{|V_{T,I}|} \sum_{f \in V_{T,I}} |f\rangle \right),$$

$$|w_a\rangle = \sum_{b \in C(a)} |w_{a,b}\rangle,$$

where

$$|w_{a,b}\rangle = \left( \sum_{u \in V_{a,b}} |u\rangle \right) \otimes \left( \frac{1}{|V_{T,I}|} \sum_{f \in V_{T,I}} |f\rangle \right).$$

Now we prove that $|w\rangle$ is a valid negative witness. First,

$$\langle w|\tau \rangle = \langle w_s|\tau \rangle + \sum_{a \in V_{T,I}} \sum_{b \in C(a)} \langle w_{a,b}|\tau \rangle = 1 + 0 = 1.$$  

Next, we show that $|w\rangle$ is orthogonal to all available input vectors on $G_c$. We deal with four kinds of available input vectors separately:

1. For any free input vector $|(s,u)\rangle = (|s\rangle - |u\rangle) \otimes (\sum_{f \in V_{T,I}} |f\rangle)$ where $u \in c^{-1}(r)$: By conditions 1 and 2 of Claim 2, $c^{-1}(r) = V_r = \bigcup_{b \in C(r)} V_{r,b}$, so we can find $b \in C(r)$ such that $u \in V_{r,b}$. Then we have $\langle w| (s,u) \rangle = 0$, since $\langle w_s| (s,u) \rangle = 1$, $\langle w_{r,b}| (s,u) \rangle = -1$, and $\langle w_{a',b'}| (s,u) \rangle = 0$ for any other $(a',b')$.

2. For any free input vector $|(v,t)\rangle = (|v\rangle - |t\rangle) \otimes |f\rangle$ where $v \in c^{-1}(f)$ for some $f \in V_{T,I}$: We have $\langle w| (v,t) \rangle = 0$, since $\langle w_s| (v,t) \rangle = 0$ for any $(a,b)$.

3. For any available input vector $|(u,v)\rangle = (|u\rangle - |v\rangle) \otimes (\sum_{f \in V_{T,I}} |f\rangle)$ where $u \in c^{-1}(a)$, $v \in c^{-1}(b)$ for some $a \in V_{T,I}$ and $b \in C(a) \cap V_{T,J}$: The availability of this input vector implies $(u,v) \in E_G$. Then, by condition 3 of Claim 2, there are two possible cases:

   - $u \in V_{a,b}, v \in V_b$: In this case, $\langle w_{a,b}| (u,v) \rangle = 1$. Also, by condition 1 of Claim 2 there exists $d \in C(b)$ such that $v \in V_{b,d}$. Then $\langle w_{b,d}| (u,v) \rangle = -1$, and $\langle w_s| (u,v) \rangle = \langle w_{a',b'}| (u,v) \rangle = 0$ for any other $(a',b')$. Thus, $\langle w| (u,v) \rangle = 0$.

   - $u \notin V_{a,b}, v \notin V_b$: In this case, we simply have $\langle w_s| (u,v) \rangle = \langle w_{a',b'}| (u,v) \rangle = 0$ for any $(a',b')$. Hence, $\langle w| (u,v) \rangle = 0$.

4. For any available input vector $|(u,v)\rangle = (|u\rangle - |v\rangle) \otimes |f\rangle$ where $u \in c^{-1}(a)$, $v \in c^{-1}(f)$ for some $a \in V_{T,J}$ and $f \in C(a) \cap V_{T,I}$: The availability of this input vector implies $(u,v) \in E_G$. Then, by condition 4 of Claim 1, we must have $u \notin V_{a,f}$. It follows that $\langle w_s| (u,v) \rangle = \langle w_{a',b'}| (u,v) \rangle = 0$ for any $(a',b')$, and hence $\langle w| (u,v) \rangle = 0$.

Now, note that $|w\rangle$ can be written as

$$|w\rangle = \sum_{u \in |s\rangle \cup V_a} \sum_{f \in V_{T,I}} \mu_{u,f} |u\rangle \otimes |f\rangle,$$

where $|\mu_{u,f}| \leq 1$ for any $(u,f)$. Meanwhile, every input vector of $\mathcal{P}$ is the sum of a constant number of some $\pm |u\rangle \otimes |f\rangle$‘s. Thus, the inner product between $|w\rangle$ and any input vector has norm $O(1)$. Since there are $O(n^2)$ input vectors in $\mathcal{P}$, the negative witness size of $\mathcal{P}$ on $G_c$ is at most $O(n^2)$.  

\[ \square \]
Combining Lemma 1 and Lemma 2 together, we know that \( \mathcal{P} \) solves the subgraph/not-a-minor problem for \( T \), and it has witness size \( O(n) \). Then, by Corollary 1 this problem can be solved by a bounded-error quantum algorithm with \( O(n) \) query complexity.

4 Time-efficient implementation

Theorem 3 implies the existence of a query-efficient quantum algorithm for the subgroup/not-a-minor problem for any tree. However, this algorithm is not necessarily time-efficient. Indeed, any span program can be evaluated by running phase estimation \([13, 18]\) on some quantum walk operator associated with this span program \([21, 8]\). But this approach does not always yield time-efficient quantum algorithms, because this quantum walk operator can be difficult to implement in general. Using some ideas from \([8]\), we nevertheless manage to implement this operator for our span program \( \mathcal{P} \) quickly and hence give a time-efficient implementation of the algorithm from Theorem 3.

**Theorem 4.** The algorithm from Theorem 3 can be implemented in \( \tilde{O}(n) \) quantum time.

**Proof.** We use the approach of \([21, 8]\) for evaluating span programs. This approach relies on an “effective” spectral gap of a quantum walk operator associated with the span program. Specifically, suppose \( Q \) is an arbitrary span program with input vectors \( |v_1\rangle, \ldots, |v_k\rangle \in \mathbb{R}^d \) and target vector \( |\tau\rangle \in \mathbb{R}^d \). Let \( \mathcal{D} \subseteq \{0,1\}^n \), and let \( W_1 = \text{wsize}_1(Q, \mathcal{D}), W_0 = \text{wsize}_0(Q, \mathcal{D}), W = \text{wsize}(Q, \mathcal{D}) \) be the positive, negative and overall witness size of \( Q \) over domain \( \mathcal{D} \), respectively. We assume that \( W_1, W_0, W, k = \text{poly}(n) \). Pick a constant \( C > \max(10, 1/W) \). Let \( \alpha = C\sqrt{W_1} \) and \( |\tau\rangle = |\tau\rangle/\alpha \). Then define

\[
V \triangleq |\tau\rangle\langle 0| + \sum_{j \in [k]} |v_j\rangle\langle j|
\]

(19)

Let \( R_A = 2\Lambda - I \), where \( \Lambda \) is the projection onto \( \text{Ker}(V) \). Moreover, for any \( x \in \mathcal{D} \), let \( R_x = 2\Pi(x) - I \), where \( \Pi(x) \triangleq \sum_{j \in [n]} |j\rangle \langle j| \), where \( I(x) \) is the index set of available input vectors on input \( x \).

The algorithm for evaluating \( Q \) work in the Hilbert space \( \mathcal{H} \triangleq \text{span}\{ |0\rangle, |1\rangle, \ldots, |k\rangle\} \). On input \( x \), it starts in \( |0\rangle \) and runs phase estimation on \( U \triangleq R_AR_x \) with precision \( 1/(10\sqrt{CW}) \) and error rate 1/10, and it accepts if and only if the measured phase is 0. This algorithm uses \( O(W) \) controlled applications of \( U \), and it correctly evaluates \( Q \) on \( x \) with probability at least 2/3. The key component of this algorithm is the implementation of \( R_A \) and \( R_x \). Usually \( R_x \) can be implemented with few input queries and polylog \( (n) \) local gates. But \( R_A \) can be much harder to implement. Let \( T_0 \) and \( T_1 \) be the time required to implement \( R_A \) and \( R_x \) respectively. Then the time complexity of this algorithm is \( \tilde{O}(W(T_0 + T_1)) \).

Now we apply this general approach to our span program \( \mathcal{P} \) for tree detection. To efficiently implement the reflection \( R_A \), we invoke the following lemma:

**Lemma 3** (Spectral lemma [25]). Let \( A \) and \( B \) be complex matrices such that they have the same number of rows and each of them has orthonormal columns. Let \( D(A,B) = A^*B \), and let \( U(A,B) = \text{Ref}_B \cdot \text{Ref}_A \). Then all the singular values of \( D(A,B) \) are at most 1. Let \( \cos \theta_1, \cos \theta_2, \ldots, \cos \theta_l \) be the singular values of \( D(A,B) \) that lie in the open interval \( (0,1) \) counted with multiplicity. Then the following is a complete list of the eigenvalues of \( U(A,B) \):

1. The \(+1\) eigenspace of \( U(A,B) \) is \( (C(A) \cap C(B)) \oplus \left( C(A)^{\perp} \cap C(B)^{\perp} \right) \);

2. The \(-1\) eigenspace of \( U(A,B) \) is \( \left( C(A) \cap C(B)^{\perp} \right) \oplus \left( C(A)^{\perp} \cap C(B) \right) \);

11
3. The other eigenvalues of \( U(A,B) \) are \( e^{2i\theta_1}, e^{-2i\theta_1}, e^{2i\theta_2}, e^{-2i\theta_2}, \ldots, e^{2i\theta_k}, e^{-2i\theta_k} \) counted with multiplicity.

Following the idea of [3], we will find two matrices \( A \) and \( B \) such that: (1) they have the same number of rows; (2) each of them has orthonormal columns; (3) \( V' \overset{\Delta}{=} A^T B = \frac{1}{\sqrt{4n}} V \). Then, Lemma [3] implies that the \(-1\) eigenspace of \( U(A,B) = \text{Ref}_B \cdot \text{Ref}_A \) is \( (C(A)^\perp \cap C(B)^\perp) \oplus (C(A)^\perp \cap C(B)) \).

Note that \( C(A)^\perp \cap C(B) = B(\text{Ker}(V')) = B(\text{Ker}(V)) \), and \( C(A)^\perp \cap C(B)^\perp \) is orthogonal to \( C(B) \). Thus, to “effectively” implement \( R_A \), we embed \( \mathcal{H} \) into \( B(\mathcal{H}) \) and treat the \(-1\) eigenspace of \( U(A,B) \) as \( \text{Ker}(V) \). That is, we simulate the behavior of \( R_A \) on any \( |\phi\rangle \in \mathcal{H} \) by the behavior of \( R_{-1} \) on \( B(|\phi\rangle) \in B(\mathcal{H}) \), where \( R_{-1} \) is the reflection about the \(-1\) eigenspace of \( U(A,B) \). This simulation is valid because \( B \) is an isometry. Now, \( R_{-1} \) can be (approximately) implemented by running phase estimation on \( U(A,B) \) and multiplying the phase by \(-1\) if the measured eigenvalue is very close to \(-1\). The precision of this phase estimation depends on the eigenvalue gap around \(-1\) of \( U(A,B) \). Let \( T_A \) and \( T_B \) be the time required to implement \( R_A \) and \( R_B \) respectively, and let \( \delta_{A,B} \) is the eigenvalue gap around \(-1\) of \( U(A,B) \). Then \( R_A \) can be implemented in time \( O((T_A + T_B)/\delta_{A,B}) \).

Next, we will describe how to factorize \( V \) into \( A \) and \( B \). But before doing that, we need to modify our span program \( \mathcal{P} \) to make it possess a more uniform structure. Specifically, we modify \( G_c^\prime \) and \( \mathcal{P} \) as follows:

1. We add dummy vertices to each block of \( G_c^\prime \), so that each block contains exactly \( n \) vertices. Then we fill in the graph with never-available edges between adjacent blocks, so that there is a complete bipartite graph between any two adjacent blocks.

2. We normalize each input vector to unit length. Specifically, for an input vector \(|(u,v)\rangle \), we scale it by a factor of \( 1/\sqrt{2|S(c(u),c(v))|} \).

3. We scale the target vector \( |\tau\rangle \) by a factor of \( 1/\sqrt{|V_{T,l}|} \) so that it has length \( \sqrt{2} \).

4. We pick a constant \( C > \max\left(10, 1/W, 1/\sqrt{|W_l|}\right) \). Let \( \alpha = C\sqrt{|W_l|} > 1 \) and

\[
|\tilde{\tau} \rangle = \frac{1}{\alpha} (|s\rangle - |t\rangle) \otimes \left( \frac{1}{\sqrt{|V_{T,l}|}} \sum_{f \in V_{T,l}} |f\rangle \right) .
\]

We add a never-available input vector

\[
|\gamma \rangle \overset{\Delta}{=} \sqrt{1 - \frac{1}{\alpha^2}} (|t\rangle - |s\rangle) \otimes \left( \frac{1}{\sqrt{|V_{T,l}|}} \sum_{f \in V_{T,l}} |f\rangle \right) .
\]

It is easy to check that this modified span program still computes the same function, and its positive and negative witness size remain \( O(1) \) and \( O(n^2) \) respectively.

Now, since each block of \( G_c^\prime \) contains \( n \) vertices, we represent the vertices and edges of \( G_c^\prime \) as follows. We denote the \( k \)-th vertex in the block \( c^{-1}(x) \) as \( (x,k) \). In particular, we denote the original \( s \) as \( (s,1) \) and denote the original \( t \) as \( (t,1) \). Then, for the edge between the vertices \( (x_1,k_1) \) and \( (x_2,k_2) \), we denote it as \( (x_1,k_1,x_2,k_2) \). But this leads to a problem. Namely, this edge also has another representation \(- (x_2,k_2,x_1,k_1) \). This could make the implementation of \( \text{Ref}_A \) and \( \text{Ref}_B \) a little harder. We solve this problem by making two copies of each input vector (except \( |\gamma\rangle \)), so that one copy corresponds to the representation \( (x_1,k_1,x_2,k_2) \) and the other corresponds to the representation \( (x_2,k_2,x_1,k_1) \). Furthermore, we make \( |\tilde{\tau}\rangle \) and \( |\gamma\rangle \) correspond to \( (s,1,t,1) \) and \( (t,1,s,1) \) respectively.
Now define

\[ I \triangleq \{ (x,k,f) : x \in V_T \cup \{s,t\}, k \in [n], f \in V_f \} \tag{22} \]

and

\[ J \triangleq \{ (x_1,k_1,x_2,k_2) : x_1,x_2 \in V_T \cup \{s,t\}, x_1 \sim x_2, \ k_1,k_2 \in [n] \}. \tag{23} \]

Then for our span program \( \mathcal{P} \), we have

\[ V = \sum_{j \in J} |v_j\rangle \langle j| \tag{24} \]

where

\[
|v_j\rangle = \begin{cases} 
\frac{1}{\sqrt{2}} (|x,k\rangle - |y,k'\rangle) \otimes \left( \frac{1}{\sqrt{\sum_{f \in \Delta_x} |f\rangle}} \sum_{f \in \Delta_x} |f\rangle \right), & \text{if } j = (x,k,y,k') \notin \{(s,1,t,1),(t,1,s,1)\}, \\
\frac{1}{\alpha} (|x,1\rangle - |t,1\rangle) \otimes \left( \frac{1}{\sqrt{|V_{T,l}|}} \sum_{f \in V_{T,l}} |f\rangle \right), & \text{if } j = (s,1,t,1), \\
\sqrt{1 - \frac{1}{\alpha^2}} (|x,1\rangle - |s,1\rangle) \otimes \left( \frac{1}{\sqrt{|V_{T,l}|}} \sum_{f \in V_{T,l}} |f\rangle \right), & \text{if } j = (t,1,s,1). 
\end{cases} \tag{25} \]

Now we are ready to give the factorization of \( V \) into \( A \) and \( B \). We will find unit vectors \( \{|a_i\} : i \in I\} \) and \( \{|b_j\} : j \in J\} \) such that

\[ \langle a_i | j \rangle \langle i | b_j \rangle = \frac{1}{\sqrt{4n}} \langle i | V | j \rangle, \tag{26} \]

Then we define

\[
A = \sum_{i \in I} (|i\rangle \otimes |a_i\rangle) \langle i |, \\
B = \sum_{j \in J} (|b_j\rangle \otimes |j\rangle) \langle j |. \tag{27} \]

It follows immediately that

\[ V' \triangleq A^\dagger B = \frac{V}{\sqrt{4n}}, \tag{28} \]

as desired. The \( |a_i\rangle \)’s are defined as follows:

- For \( i = (x,k,f) \), where \( x \notin \{s,t\} \) or \( k \neq 1 \), let

  \[ |a_i\rangle = \frac{1}{\sqrt{4n}} \sum_{y \in M(x,f)} \sum_{k' \in [n]} (|x,k,y,k'\rangle + |y,k',x,k\rangle). \tag{29} \]

  Note that since \( |M(x,f)| = 2 \), \( |a_i\rangle \) is indeed a unit vector.

- For \( i = (s,1,f) \), let

  \[
  |a_i\rangle = \frac{1}{\sqrt{4n}} \left( \sum_{r \in [n]} (|s,1,r,k\rangle + |r,k,s,1\rangle) + \sum_{k=2}^{n} (|s,1,t,k\rangle + |t,k,s,1\rangle) \right) \\
  + \frac{1}{\sqrt{2n}} \left( \frac{1}{\alpha} |s,1,t,1\rangle + \sqrt{1 - \frac{1}{\alpha^2}} |t,1,s,1\rangle \right). \tag{30} \]
For $i = (t, 1, f)$, let
\[ |a_i\rangle = \frac{1}{\sqrt{4n}} \left( \sum_{k \in [n]} (|t, 1, f, k\rangle + |f, k, t, 1\rangle) + \sum_{k=2}^{n} (|t, 1, s, k\rangle + |s, k, t, 1\rangle) \right) 
+ \frac{1}{\sqrt{2n}} \left( \frac{1}{\alpha} |s, 1, t, 1\rangle + \sqrt{1 - \frac{1}{\alpha^2}} |t, 1, s, 1\rangle \right). \] (31)

The $|b_j\rangle$’s are defined as follows:

- For $j = (x_1, k_1, x_2, k_2)$, let
\[ |b_j\rangle = \frac{1}{\sqrt{2|S_{x_1,x_2}|}} \sum_{f \in S_{x_1,x_2}} (|x_1, k_1, f\rangle - |x_2, k_2, f\rangle). \] (32)

It is easy to check that these $|a_i\rangle$’s and $|b_j\rangle$’s are indeed unit vectors and they satisfy Eq.(26), as promised.

Now we describe the implementation of $R_x$ and $R_A$. We embed the space $\mathcal{H} = \text{span} \{ |j\rangle : j \in J \}$ into $B(\mathcal{H}) = \text{span} \{ |b_j\rangle | j \rangle : j \in J \}$. To implement $R_x$, we detect $j$ in the second register and multiply the phase by $-1$ if $|v_j\rangle$ is an unavailable input vector on $x$ (this can be tested by querying the edge labelled by $j$). To implement $R_A$, we run phase estimation on the quantum walk operator $U(A,B) = \text{Ref}_B \cdot \text{Ref}_A$ with precision $\delta_{AB}/3$ and multiply the phase by $-1$ if the measured eigenvalue is $(\delta_{AB}/3)$-close to $-1$. It remains to analyze the spectral gap $\delta_{AB}$ around $-1$ of $U(A,B)$ and give explicit implementation of Ref$_A$ and Ref$_B$.

**Lemma 4.** The eigenvalue gap $\delta_{AB}$ around $-1$ of $U(A,B) = \text{Ref}_B \cdot \text{Ref}_A$ is $\Omega(1)$.

**Proof.** By Lemma 3, it is sufficient to show that the singular value gap around $0$ of $V' = A^\dagger B$ is $\Omega(1)$. To prove this, it is sufficient to show that the smallest non-zero eigenvalue of
\[ \Delta \triangleq V'V'^\dagger = \frac{1}{4n} \sum_{j \in J} |v_j\rangle \langle v_j| \] (33)

is $\Omega(1)$. Let $F = \{ (x,y) : x,y \in V_T \cup \{s,t\}, x \sim y \}$, and let $F(x,y) = \{ (x,k,y,k') : k,k' \in [n] \}$ for any $(x,y) \in F$. Then we can write $\Delta$ as
\[ \Delta = \sum_{(x,y) \in F} \Delta_{x,y} \] (34)

where
\[ \Delta_{x,y} \triangleq \frac{1}{4n} \sum_{j \in F(x,y)} |v_j\rangle \langle v_j| \] (35)

Now, for any $j = (x,k,y,k')$, where $(x,y) \not\in \{(s,t),(t,s)\}$, we have
\[ |v_j\rangle \langle v_j| = \frac{1}{2|S_{x,y}|} \sum_{f,f' \in S_{x,y}} (|x,k,f\rangle \langle x,k,f'| + |y,k,f\rangle \langle y,k,f'|) 
- \frac{1}{2|S_{x,y}|} \sum_{f,f' \in S_{x,y}} (|x,k,f\rangle \langle y,k',f'| + |y,k',f'\rangle \langle x,k,f|). \] (36)

Taking the sum of Eq.(36) over $k,k' \in [n]$ yields
\[ \Delta_{x,y} = A_{x,y} \otimes I_n + \frac{1}{n} B_{x,y} \otimes E_n, \] (37)
where $I_n = \sum_{k \in [n]} |k\rangle\langle k|$ and $E_n = \sum_{k,k' \in [n]} |k\rangle\langle k'|$, and $A_{x,y}$ and $B_{x,y}$ are some matrices independent of $n$ (here we have switched the order of $k$-register and $f$-register). This holds for any $(x, y) \notin \{(s, t), (t, s)\}$.

On the other hand, we also have

$$\Delta_x + \Delta_y = \tilde{A} \otimes I_n + \frac{1}{n} \tilde{B} \otimes E_n,$$

(38)

where $\tilde{A}$ and $\tilde{B}$ are some matrices independent of $n$. This can be derived from the fact that for any $j = (s, t, k')$ or $(t, k', s, k)$, where $(k, k') \neq (1, 1)$,

$$|v_j\rangle = \frac{1}{2|V_T|} \sum_{f, f' \in V_T} (|s, k, f\rangle\langle s, k, f'| + |t, k, f\rangle\langle t, k, f'|)$$

and the fact that

$$|v_{(s,1,t,1)}\rangle\langle v_{(s,1,t,1)}| + |v_{(t,1,s,1)}\rangle\langle v_{(t,1,s,1)}| = \frac{1}{|V_T|} \sum_{f, f' \in V_T} (|s, 1, f\rangle\langle s, 1, f'| + |t, 1, f\rangle\langle t, 1, f'|)$$

$$- \frac{1}{|V_T|} \sum_{f, f' \in V_T} (|s, 1, f\rangle\langle s, 1, f'| + |t, 1, f\rangle\langle t, 1, f'|).$$

(40)

Now by Eqs. (37) and (38) we obtain

$$\Delta = A \otimes I_n + \frac{1}{n} B \otimes E_n,$$

(41)

where $A$ and $B$ are some matrices independent of $n$. Then, some simple linear algebra shows that the spectrum of $\Delta$ (i.e. the set of eigenvalues sans multiplicity) does not depend on $n$. In particular, the smallest non-zero eigenvalue of $\Delta$ is $\Omega(1)$, as desired.

**Lemma 5.** Ref$_A$ can be implemented in polylog$(n)$ time.

**Proof.** Since Ref$_A$ is the reflection about the span of $|i\rangle \otimes |a_i\rangle$’s, we can implement it as $U_A O_A U_A^\dagger$, where $U_A$ is any unitary operation that transforms $|i\rangle \otimes \bar{0}$ to $|i\rangle \otimes |a_i\rangle$, and $O_A$ is the reflection about $\bar{0}$ on the second subsystem. Obviously, $O_A$ can be implemented in polylog$(n)$ time. So it remains to show that $U_A$ can be implemented in polylog$(n)$ time.

Observe that $|a_i\rangle$ can be written as:

$$|a_{(x,k,f)}\rangle = \frac{1}{\sqrt{4n}} \sum_{y \in M(x,f)} \sum_{k' \in [n]} Q (|x, k, y, k'\rangle + |y, k', x, k\rangle),$$

(42)

where $Q$ is a unitary operation acting on the span of $|x, k, y, k'\rangle$’s such that

$$Q |x, k, y, k'\rangle = |x, k, y, k'\rangle,$$

$$\forall (x, k, y, k) \notin \{(s, 1, t, 1), (t, 1, s, 1)\};$$

$$Q \left( \frac{1}{\sqrt{2}} |s, 1, t, 1\rangle + \frac{1}{\sqrt{2}} |t, 1, s, 1\rangle \right) = \frac{1}{\alpha} |s, 1, t, 1\rangle + \sqrt{1 - \frac{1}{\alpha^2}} |t, 1, s, 1\rangle.$$

(43)
So we can generate \( |i \rangle \otimes |a_i \rangle \) from \( |i \rangle \otimes |0 \rangle \) as follows:

\[
|x, k, f \rangle \otimes |0, 0, 0, 0 \rangle \rightarrow |x, k, f \rangle \otimes |x, k, 0, 0 \rangle \\
\rightarrow |x, k, f \rangle \otimes \frac{1}{\sqrt{n}} \sum_{k' \in [n]} |x, k, 0, k' \rangle \\
\rightarrow |x, k, f \rangle \otimes \frac{1}{\sqrt{2n}} \sum_{y \in M(x,f)} \sum_{k' \in [n]} |x, k, y, k' \rangle \\
\rightarrow |x, k, f \rangle \otimes \frac{1}{\sqrt{4n}} \sum_{y \in M(x,f)} \sum_{k' \in [n]} Q(|x, k, y, k' \rangle + |y, k', x, k \rangle) \\
\rightarrow |x, k, f \rangle \otimes \frac{1}{\sqrt{n}} \sum_{y \in M(x,f)} \sum_{k' \in [n]} Q(|x, k, y, k' \rangle + |y, k', x, k \rangle) \\
= |x, k, f \rangle \otimes |a_{(x,k,f)} \rangle,
\]

where

- The first step is accomplished by performing a unitary operation \( U_{A,1} \) that transforms \( |x, k, 0, 0 \rangle \) to \( |x, k, x, k \rangle \) on the first, second, fourth and fifth registers;
- The second step is accomplished by performing a unitary operation \( U_{A,2} \) that transforms \( |0 \rangle \) to \( |x, f \rangle \otimes \frac{1}{\sqrt{2}} \sum_{|y \rangle \in M(x,f)} |y \rangle \) on the last register;
- The third step is accomplished by performing a unitary operation \( U_{A,3} \) that transforms \( |x, f \rangle \otimes |0 \rangle \) to \( |x, f \rangle \otimes \frac{1}{\sqrt{2}} \sum_{|y \rangle \in M(x,f)} |y \rangle \) on the first, third and sixth registers;
- The fourth step is accomplished by performing a unitary operation \( U_{A,4} \) that transforms \( |x, k \rangle \otimes |x, k, y, k' \rangle \) to \( |x, k \rangle \otimes \frac{1}{\sqrt{2}} (|x, k, y, k' \rangle + |y, k', x, k \rangle) \) on all but the third registers;
- The last step is accomplished by performing \( Q \) on the last four registers.

Formally, let \( U_A = QU_{A,4}U_{A,3}U_{A,2}U_{A,1} \). Clearly, all of \( U_{A,1}, U_{A,2}, U_{A,3}, U_{A,4} \) and \( Q \) can be implemented in polylog \( n \) time, and hence so is \( U_A \).

**Lemma 6.** Ref \( B \) can be implemented in polylog \( n \) time.

**Proof.** Since \( \text{Ref}_B \) is the reflection about the span of \( |b_j \rangle \otimes |a_j \rangle \)'s, we can implement it as \( U_BO_BU_B^\dagger \), where \( U_B \) is any unitary operation that transforms \( |0 \rangle \otimes |j \rangle \) to \( |b_j \rangle \otimes |j \rangle \), and \( O_B \) is the reflection about \( |0 \rangle \) on the first subsystem. Obviously, \( O_B \) can be implemented in polylog \( n \) time. So it remains to show that \( U_B \) can be implemented in polylog \( n \) time.

We generate \( |b_j \rangle \otimes |j \rangle \) from \( |0 \rangle \otimes |j \rangle \) as follows:

\[
|0, 0, 0 \rangle \otimes |x_1, k_1, x_2, k_2 \rangle \rightarrow |0, 0 \rangle \otimes \left( \frac{1}{\sqrt{|S_{x_1,x_2}|}} \sum_{f \in S_{x_1,x_2}} |f \rangle \right) \otimes |x_1, k_1, x_2, k_2 \rangle \\
\rightarrow \frac{1}{\sqrt{2}} (|x_1, k_1 \rangle - |x_2, k_2 \rangle) \otimes \left( \frac{1}{\sqrt{|S_{x_1,x_2}|}} \sum_{f \in S_{x_1,x_2}} |f \rangle \right) \otimes |x_1, k_1, x_2, k_2 \rangle \\
= |b_{(x_1,k_1,x_2,k_2)} \rangle \otimes |x_1, k_1, x_2, k_2 \rangle.
\]

where

- The first step is accomplished by performing a unitary operation \( U_{B,1} \) that transforms \( |0 \rangle \otimes |x_1, x_2 \rangle \) to \( \left( \frac{1}{\sqrt{|S_{x_1,x_2}|}} \sum_{f \in S_{x_1,x_2}} |f \rangle \right) \otimes |x_1, x_2 \rangle \) on the third, fourth and sixth registers.
The second step is accomplished by performing a unitary operation $U_{B,2}$ that transforms

$|0,0\rangle \otimes |x_1,k_1,x_2,k_2\rangle$ to $\frac{1}{\sqrt{2}}(|x_1,k_1\rangle - |x_2,k_2\rangle) \otimes |x_1,k_1,x_2,k_2\rangle$ on the first two and last four registers.

Formally, let $U_B = U_{B,2}U_{B,1}$. Clearly, both $U_{B,1}$ and $U_{B,2}$ can be implemented in $\text{polylog}(n)$ time, and hence so is $U_B$.

Combining Lemma 4, Lemma 5 and Lemma 6, we know that $R_\Lambda$ can be implemented in $\text{polylog}(n)$ time. Since $R_x$ can be also implemented in $\text{polylog}(n)$ time, $U = R_\Lambda R_x$ can be implemented in $\text{polylog}(n)$ time. Then, since $\mathcal{P}$ has witness size $O(n)$, the time complexity of our algorithm is $\tilde{O}(n)$, as claimed.

5 Conclusion and open problems

To summarize, we have presented a time-efficient span-program-based quantum algorithm for the subgraph/not-a-minor problem for any tree. This algorithm has $O(n)$ query complexity, which is optimal, and $\tilde{O}(n)$ time complexity, which is tight up to polylog factors.

Our work raises many interesting questions:

- As mentioned in Section 3, our span program $\mathcal{P}$ accepts some graphs that contain $T$ as a minor but not as a subgraph. Thus, it fails to solve the $T$-containment problem. However, it is not true that $\mathcal{P}$ accepts every graph containing $T$ as a minor. One can see that a graph must possess certain structure in order to be accepted by $\mathcal{P}$. But this structure is not easy to characterize. It would be interesting to know exactly what kind of graphs are accepted by $\mathcal{P}$. If we have a better understanding of this matter, we might be able to modify $\mathcal{P}$ to make it reject any graph that does not contain $T$ as a subgraph, thus solving the $T$-containment problem.

- In this paper, we have focused on the detection of trees. It is worth studying the detection of more complicated patterns, such as cycles and cliques. In particular, can we design time-efficient span-program-based quantum algorithms for those problems as well?

- Perhaps the most interesting direction is to investigate the potential of our “parallel-flow” technique for designing span programs. In particular, can we use this technique to improve learning graphs [5]? The basic idea of learning graph is to run a single flow from a vertex (i.e. the empty set) to some vertices (i.e. those contain a 1-certificate) on an exponentially large graph, and its efficiency is determined by the energy of this flow. We observe that many decision problems can be decomposed into several correlated subproblems, so that an input is a positive instance of the original problem if and only if it is a positive instance of all the subproblems simultaneously. Perhaps we can enhance the efficiency of learning graphs by dividing the original flow into several parallel flows, so that each flow corresponds to one subproblem. We might also need to make these flows interfere somehow in order to respect the correlations among these subproblems. It is possible that the total energy of these flows might be smaller than that of the original flow. Can we formalize this idea and use it to improve previous learning-graph-based quantum algorithms?
Acknowledgement

We thank Ben Reichardt and Umesh Vazirani for helpful discussion on this topic. This research was supported by NSF Grant CCR-0905626 and ARO Grant W911NF-09-1-0440.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844-856, July 1995.

[2] A. Ambainis. Quantum walk algorithm for element distinctness. In *Proc. 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22-31, 2004.

[3] A.T. Balaban. *Chemical applications of graph theory*. Academic Press, 1976.

[4] A. Belovs. Span-program-based quantum algorithm for the rank problem. 2011, arXiv:1103.0842.

[5] A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proc. 44th ACM Symposium on Theory of Computing (STOC)*, pages 77–84, 2012.

[6] A. Belovs. Learning-graph-based Quantum Algorithm for k-distinctness. In *Proc. 53th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 207-216, 2012.

[7] A. Belovs and T. Lee. Quantum algorithm for *k*-distinctness with prior knowledge on the input. 2011, arXiv:1108.3022.

[8] A. Belovs and B. W. Reichardt. Span programs and quantum algorithms for st-connectivity and claw detection. In *Proc. 20th European Symposia on Algorithm (ESA)*, pages 193-204, 2012.

[9] A. M. Childs and R. Kothari. Quantum query complexity of minor-closed graph properties. In *Proc. 28th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *Leibniz International Proceedings in Informatics*, pages 661–672, 2011.

[10] D. Gavinsky and T. Ito. A quantum query algorithm for the graph collision problem. 2012, arXiv:1204.1527.

[11] R. Heckel. Graph transformation in a nutshell. *Electronic Notes in Theoretical Computer Science* 148, pp. 187198, 2006.

[12] S. Jeffery, R. Kothari, and F. Magniez. Nested Quantum Walks with Quantum Data Structures. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA 13)*, pages 1474-1485, 2013.

[13] A. Y. Kitaev. Quantum measurements and the Abelian Stabilizer Problem. arXiv:quant-ph/9511026.

[14] M. Karchmer, and A. Wigderson. On span programs. In *Proc. 8th IEEE Symp. Structure in Complexity Theory*, pages 102–111, 1993.

[15] T. Lee, F. Magniez, and M. Santha. A learning graph based quantum query algorithm for finding constant-size subgraphs. *Chicago Journal of Theoretical Computer Science*, 2012.
A Proof of Claim 2

For convenience, we introduce the following notation. Let $H = (V_H, E_H)$ be an arbitrary graph. For any $u \in V_H$, we say that $(H, u)$ is good with respect to $T$ if there exists a subgraph $H' = (V_{H'}, E_{H'})$ of $H$ and $E' \subseteq E_{H'}$ such that $E' H'$ is isomorphic to $T$ and the root of $E' H'$ contains $u$ (and thus $u$ must be involved in $E'$).

We will prove the following claim (which is stronger than Claim 2):

Claim 3. For any graph $G = (V_G, E_G)$ and coloring $c : V_G \rightarrow V_T$, there exist $L \in \mathbb{N}$, $W \subseteq c^{-1}(r)$, $\{V_{a,i} \subseteq c^{-1}(a) : a \in V_{T,i}, l \in [L]\}$ and $\{V_{a,b,l} \subseteq c^{-1}(a) : a \in V_{T,i}, b \in C(a), l \in [L]\}$, such that:

1. $\forall a \in V_{T,i}, \forall l \in [L], V_{a,l}$ is the disjoint union of $V_{a,b,l}$'s for $b \in C(a)$;
2. $c^{-1}(r)$ is the disjoint union of $W$ and $V_{i,l}$'s for $l \in [L]$;
3. \( \forall a \in V_{T,i}, \forall b \in C(a) \cap V_{T,i}, \forall l \in [L], N_{a\rightarrow b}(V_{a,b,i}) \subseteq V_{b,i} \) and \( N_{b\rightarrow a}(V_{b,i}) \subseteq V_{a,b,i} \);

4. \( \forall a \in V_{T,i}, \forall b \in C(a) \cap V_{T,i}, \forall l \in [L], N_{a\rightarrow b}(V_{a,b,i}) = \emptyset \);

5. Let \( U_l = \cup_{a \in V_{a,b}} V_{a,b} \ \forall l \in [L] \), and let \( U = \cup_{l \in [L]} U_l \). Then \( U_1, U_2, \ldots, U_L \) are the vertex sets of the connected components of \( G_{c|U} \);

6. Let \( Z = V_G \setminus U \). Then \( \forall w \in W, (G_c|Z,w) \) is good with respect to \( T \).

Let us first show that Claim 3 indeed implies Claim 2. Suppose \( G_c \) does not contain \( T \) as a minor. By applying Claim 3, we obtain the \( W, V_{a,l}'s \) and \( V_{a,b,l}'s \) satisfying the above conditions. Now define \( V_a = \cup_{l \in [L]} V_{a,l} \) and \( V_{a,b} = \cup_{l \in [L]} V_{a,b,l} \), for any \( a \in V_{T,i} \) and \( b \in C(a) \). We claim that these \( V_a \)'s and \( V_{a,b} \)'s satisfy all the conditions of Claim 2. This is because:

- Since the \( V_{a,l}'s \) and \( V_{a,b,l}'s \) satisfy conditions 1, 3 and 4 of Claim 3, and the \( V_r \)'s or \( V_{a,b} \)'s are simply the union of the \( V_{a,l}'s \) or \( V_{a,b,l}'s \) respectively, it is obvious that the \( V_r \)'s and \( V_{a,b} \)'s satisfy conditions 1, 3 and 4 of Claim 2.

- Since \( G_c \) does not contain \( T \) as a minor, by condition 4 of Claim 3, we must have \( W = \emptyset \). Then, by condition 2 of Claim 3 and \( V_r = \cup_{l \in [L]} V_{r,l} \), we have \( V_r = c^{-1}(r) \). So condition 2 of Claim 2 is also fulfilled.

Now it remains to prove Claim 3.

Proof of Claim 3. Proof by induction on the depth of \( T \).

1. Basis: Suppose \( T \) has depth 0. Namely, \( T \) contains only a root node \( r \). Then we simply let \( W = c^{-1}(r) = V_G \) and \( L = 0 \) (or let \( V_{r,l} = \emptyset \) for any \( l \)). Then all the conditions of Claim 3 are trivially satisfied.

Suppose \( T \) has depth 1. Namely, \( T \) contains a root node \( r \) and its children \( f_1, f_2, \ldots, f_k \) (which are the leaves of \( T \)). We build the desired \( W, V_{a,l}'s \) and \( V_{a,b,l}'s \) as follows:

- Initially, set \( W \leftarrow \emptyset \) and \( L \leftarrow 0 \).
- For each vertex \( u \in c^{-1}(r) \), do:
  - If \( N_{r \rightarrow f_j}(u) \neq \emptyset \) for every \( j \in [k] \), then set \( W \leftarrow W \cup \{u\} \);
  - Otherwise, there exists some \( j \in [k] \) such that \( N_{r \rightarrow f_j}(u) = \emptyset \). Then:
    - Set \( L \leftarrow L + 1 \);
    - Set \( V_{r,L} \leftarrow \{u\} \) and \( V_{r,f_j,L} \leftarrow \{u\} \);
    - Set \( V_{r,f_{j'},L} \leftarrow \emptyset \) for any \( j' \neq j \).

Now we show that the \( W, V_{a,l}'s \) and \( V_{a,b,l}'s \) obtained by this algorithm satisfy all the conditions of Claim 3:

- By the construction, it is obvious that \( V_{r,l} \) is the disjoint union of \( V_{r,f_{j,l}}'s \) for \( j \in [k] \), for any \( l \). So condition 1 is fulfilled;
- Since each vertex in \( c^{-1}(r) \) is put into either \( W \) or some \( V_{r,l} \), we know that \( c^{-1}(r) \) is the disjoint union of \( W \) and the \( V_{r,l}'s \). So condition 2 is also fulfilled;
- Since there is only one internal node \( r \) and all of its children are leaves, the situation described by condition 3 does not exist and hence condition 3 is automatically satisfied;
• By the definition of \(V_{r,f_i}'s\) and \(V_{r,f_j}'s\), we have \(N_{r \rightarrow f_j}(V_{r,f_j}) = \emptyset\). So condition 4 is also satisfied;

• Since each \(U_i\) contains only one vertex, \(G_c|_U\) is simply a collection of isolated vertices, and hence \(U_1, U_2, \ldots, U_{l_j}\) are the vertex sets of the connected components of \(G_c|_U\). Therefore, condition 5 is fulfilled;

• For any \(w \in W\), it has a neighbor in each of \(c^{-1}(f_1), c^{-1}(f_2), \ldots, c^{-1}(f_k)\), and hence \(G_c|Z\) contains a tree that is isomorphic to \(T\) and its root is \(w\). So \((G_c|Z, w)\) is good with respect to \(T\). Thus, condition 6 is fulfilled.

2. **Inductive step:** Suppose that Claim 3 holds for any graph \(G'\) and coloring \(c'\) with respect to any tree \(T'\) of depth at most \(d\).

Let \(T\) be a tree of depth \(d + 1\). Suppose its root \(r\) has \(k\) children \(d_1, d_2, \ldots, d_k\). For each \(j \in [k]\), since \(T_{d_j}\) has depth at most \(d\), we can apply Claim 3 to \(G_{d_j}\) with respect to \(T_{d_j}\), and obtain the \(W_j, V_{a,b,j}'s\) and \(V_{a,b,j}'s\) (where \(a \in V_{d_j}, b \in C(a)\) and \(\beta \in [L_j]\) for some \(L_j\)) satisfying the conditions of Claim 3. In particular, let \(U_j' = \bigcup_{\alpha \in V_{d_j}} V_{a,b,\alpha}^j\) for \(\beta \in [L_j]\), and let \(U_j = \bigcup_{\beta \in [L_j]} U_j'\) for \(j \in [k]\). Then \(U_1', U_2', \ldots, U_k'\) are the vertex sets of the connected components of \(G_{d_j}\), which is a subgraph of \(G_{d_j}\), for any \(j \in [k]\). Let \(Q_j = c^{-1}(r) \cup U_j\) for \(j \in [k]\), and let \(Q = \bigcup_{j \in [k]} Q_j\). Then let \(H' = G_c|_{Q_j}\), and let \(H = G_c|_Q\). Note that each \(H_j\) is a subgraph of \(H\).

Now consider the connected components of \(H\) and \(H_j's\). Suppose the vertex sets of the connected components of \(H\) are \(A_1, A_2, \ldots, A_m\), and the vertex sets of the connected components of \(H_j\) are \(B_1, B_2, \ldots, B_{m_j}\) for some \(m_j\) for \(j \in [k]\). Note that each \(A_i\) is the union of some \(B_i\)'s (for different \((j, t)\)'s), while each \(B_i\) is the union of several \(U_i'\)'s (for different \(\beta\)'s) and some subset of \(c^{-1}(r)\). Let \(E_{i,j} = \{t \in [m_j] : B_i \subseteq A_t\}\), and let \(F_{j,t} = \{\beta \in [L_j] : U_i' \subseteq B_i\}\), for \(i \in [m], j \in [k]\) and \(t \in [m_j]\). Note that \(A_i \cap c^{-1}(r) = \bigcup_{t \in E_{i,j}} \left(B_i \cap c^{-1}(r)\right)\) for any \(i, j\).

We build the desired \(W, V_{a,l}'s\) and \(V_{a,b,l}'s\) as follows:

• Initially, set \(W \leftarrow \emptyset\) and \(L \leftarrow 0\).

• For \(i := 1\) to \(m\) do:
  
  (a) If \(N_{r \rightarrow d_j}(A_i \cap c^{-1}(r)) \cap W_j \neq \emptyset \) for every \(j \in [k]\), then set \(W \leftarrow W \cup (A_i \cap c^{-1}(r))\);
  
  (b) Otherwise, there exists some \(j \in [k]\) such that \(N_{r \rightarrow d_j}(A_i \cap c^{-1}(r)) \cap W_j = \emptyset\). Then:

  For each \(t \in E_{i,j}\) do:
  
  - Set \(L \leftarrow L + 1\);
  
  - Set \(V_{a,l} \leftarrow B_i \cap c^{-1}(r)\) and \(V_{r,\alpha,l} \leftarrow B_i \cap c^{-1}(r)\);
  
  - Set \(V_{a,B} \leftarrow \bigcup_{\beta \in F_{j,t}} V_{a,b,\alpha}\) and \(V_{a,b,l} \leftarrow \bigcup_{\beta \in F_{j,t}} V_{a,b,\alpha}\), for any \(a \in V_{d_j}^j\) and \(b \in C(a)\);
  
  - Set any other \(V_{a,l}\) or \(V_{a,b,l}\) to be \(\emptyset\).

Now we show that the \(W, V_{a,l}'s\) and \(V_{a,b,l}'s\) obtained by the above algorithm satisfy all the conditions of Claim 3

• To prove that the \(V_{a,l}'s\) and \(V_{a,b,l}'s\) satisfy conditions 1, 3 and 4, we consider two possible cases separately:

  - \(a \neq r\): Since the \(V_{a,b,\alpha}'s\) and \(V_{a,b,\alpha}'s\) satisfy the conditions 1, 3 and 4 for each \(j \in [k]\) (by the inductive hypothesis), and the \(V_{a,l}'s\) or \(V_{a,b,l}'s\) are simply the union of several
$V_{d,j}$'s or $V_{a,b,j}$'s (for consistent choice of $j$'s and $\beta$'s) respectively, it is easy to see that the $V_{a,j}$'s and $V_{a,b,l}$'s also satisfy conditions 1, 3 and 4.

- **a = r:**
  
  (a) By construction, for any $l$, exactly one of the $V_{r,d,l}$'s (for $j \in [k]$) equals $V_{r,l}$, and the other $V_{r,d,l}$'s are all empty. So $V_{r,l}$ is indeed the disjoint union of the $V_{r,d,l}$'s (for $j \in [k]$). Hence, condition 1 is satisfied.

  (b) For any internal node $d_j$, for any $l$, we have three possible cases:
  
  * $V_{r,d,l} = V_{d,l} = \emptyset$;
  
  * $V_{r,d,l} = B_{l} \cap c^{-1}(r)$ and $V_{d,l} = \bigcup_{\beta \in F_j} V_{d,j,\beta}$ for some $t \in [m]$;
  
  For the first case, we have $N_{r-d,l}(V_{r,d,l}) = V_{d,l} = \emptyset$ and $N_{d,l-t}(V_{d,l}) = V_{r,d,l} = \emptyset$. For the second case, since $B_{l}$ is a connected component of $H^l$, we also have $N_{r-d,l}(V_{r,d,l}) \subseteq V_{d,l}$ and $N_{d,l-t}(V_{d,l}) \subseteq V_{r,d,l}$. Therefore, condition 3 is fulfilled.

  (c) For any leaf $d_j$, we have $W_j = c^{-1}(d_j)$. So if $N_{r-d,l}(A_i \cap c^{-1}(r)) \cap W_j = \emptyset$, then $N_{r-d,l}(A_i \cap c^{-1}(r)) = \emptyset$. This implies that $N_{r-d,l}(V_{r,d,l}) = \emptyset$ for any $l$. Hence, condition 4 is also satisfied.

- **Since $c^{-1}(r) = \bigcup_{i \in [m]} (A_i \cap c^{-1}(r))$ and $A_i \cap c^{-1}(r) = \bigcup_{j \in E_i} (B_{l,j} \cap c^{-1}(r))$ for any $i,j$, by the construction of $W$ and the $V_{r,l}$'s, we know that they form a partition of $c^{-1}(r)$. Thus, condition 2 is also fulfilled.**

- **Let $U_l = \bigcup_{a \in V_j} V_{a,l}$ for $l \in [L]$, and let $U = \bigcup_{l \in [L]} U_l$. Recall that $U_{l1}, U_{l2}, \ldots, U_{lJ}$ are the vertex sets of the connected components of $G_{c\mid U}$ for any $j \in [k]$ (by the inductive hypothesis). But viewing from the bigger graph $H^l$, different $U_{l1}$ and $U_{l2}$ might become connected via some vertex in $c^{-1}(r)$. Now each $B_{l,j}$ is the vertex set of a connected component of $H^l$, so it is the union of some subset of $c^{-1}(r)$ and some $U_{l,j}$'s (for different $\beta$'s). Then, by construction, we have:**

  - For any $l$, $U_l = B_{l,j}$ for some $j$ and $t$. So the vertices in $U_l$ are connected in $G_{c\mid U}$;
  
  - For any $l \neq l'$, $U_l$ and $U_{l'}$ are disjoint;
  
  - For any $l \neq l'$, $U_l$ and $U_{l'}$ do not share any vertex in $c^{-1}(r)$. This is because:
    
    * For any $i \neq l'$, $A_i$ and $A_{l'}$ do not share any vertex in $c^{-1}(r)$, because $A_i$ and $A_{l'}$ are the vertex sets of different connected components of $H$. It follows that, for any $i \neq l'$, $t \in E_{i,j}$ and $t' \in E_{l',j}$, $B_{l,j}$ and $B_{l',j}$ do not share any vertex in $c^{-1}(r)$, since $B_{l,j} \subseteq A_i$ and $B_{l',j} \subseteq A_{l'}$;
    
    * Also, for any $l \neq l'$, $B_{l,j}$ and $B_{l',j}$ do not share any vertex in $c^{-1}(r)$, since $B_{l,j}$ and $B_{l',j}$ are the vertex sets of different connected components of $H$;
    
    * Now, for $l \neq l'$, we have $U_l \cap c^{-1}(r) = B_{l,j} \cap c^{-1}(r)$ for some $j$ and $t$, and $U_{l'} \cap c^{-1}(r) = B_{l',j} \cap c^{-1}(r)$ for some $j'$ and $t'$. There are two possible cases: (1) either $t \in E_{i,j}$ and $t' \in E_{i',j'}$, when $i \neq l'$; (2) or $t \in E_{i,j}$ and $t \neq t'$. Either way, the above facts imply that $U_l$ and $U_{l'}$ do not share any vertex in $c^{-1}(r)$.

  - For any $(j, \beta) \neq (j', \beta')$, there is no edge between $U^l_{\beta}$ and $U^l_{\beta'}$ in $G_{c\mid U}$;

  - The above facts imply that $U_l$ and $U_{l'}$ are disconnected in $G_{c\mid U}$ for any $l \neq l'$.

  Combining these facts, we know that $U_1, U_2, \ldots, U_L$ are the vertex sets of the connected components of $G_{c\mid U}$. So condition 5 is fulfilled.

- **Let $Z_j = U^j \setminus U^l$, for $j \in [k]$. Note that $Z_j \cap U = \emptyset$. Then, by the inductive hypothesis, for any $w \in W^l$, $(G_{c\mid Z_j}, w)$ is good with respect to $T^d_j$.**
Now consider any $w \in W$. Since $c^{-1}(r) = \bigcup_{i \in [m]} (A_i \cap c^{-1}(r))$, there exists $i \in [m]$ such that $w \in A_i$. Then by construction, we must have $(A_i \cap c^{-1}(r)) \subseteq W$, and hence $A_i \cap V_{i,l} = \emptyset$ for any $l$, and hence $A_i \cap U = \emptyset$.

Pick any $w^j \in N_{r \to d_j}(A_i \cap c^{-1}(r)) \cap W^j$, for $j \in [k]$. Since $(G_c|Z^j, w^j)$ is good with respect to $T^{d_j}$, $G_c|Z^j$ contains a subgraph $\tilde{G}_j$ such that $\tilde{G}_j$ can be contracted into a tree $\tilde{T}^j$ which is isomorphic to $T^{d_j}$ and the root of $\tilde{T}^j$ contains $w^j$. Importantly, the vertices in $c^{-1}(r)$ and $U^j$ are not involved in such contractions (because they are not in $Z'$). Do such contractions for each $j \in [k]$.

Meanwhile, since $A_i$ is a connected component of $H$, we can contract it into a single vertex. This contraction can be performed simultaneously with the above contractions. The reason is that $A_i$ contains only some vertices in $c^{-1}(r)$ and $U^j$’s, which are not involved in any of the above contractions. Thus, even after the above contractions, we can still contract $A_i$ into a single vertex $v_i \triangleq A_i$ which contains $w$. Also, $v_i$ will be connected to $w^1, w^2, \ldots, w^k$, which are the roots of $\tilde{T}^1, \ldots, \tilde{T}^k$ which are isomorphic to $T^{d_1}, T^{d_2}, \ldots, T^{d_k}$ respectively. Thus, the resulting graph contains a tree isomorphic to $T$. Fig[5] illustrates an example of such transformations.

Now since $A_i \cap U = \emptyset$ and $Z^j \cap U = \emptyset$ for any $j \in [k]$, the above transformation involves only some vertices in $Z = V_G \setminus U$. Hence, $(G_c|Z, w)$ is good with respect to $T$. So condition 6 is also fulfilled.

\[\square\]
Figure 5: In this example, $T$’s root $r$ has 3 children $d_1, d_2, d_3$. For the given $G_c$, we have $W^1 = \{u_4, u_5\}$, $U^1 = \{u_6\}$, $W^2 = \{u_7\}$, $U^2 = \{u^8\}$, $W^3 = \{u_{11}\}$, $U^3 = \{u_9, u_{10}, u_{19}\}$, $A_1 = \{u_1, u_2, u_3, u_6, u_9, u_{10}, u_{19}\}$ and $A_2 = \{u_8\}$. Note that $N_{r \rightarrow d_1}(A_1 \cap c^{-1}(r)) \cap W^1 = \{u_4\}$, $N_{r \rightarrow d_2}(A_1 \cap c^{-1}(r)) \cap W^2 = \{u_7\}$ and $N_{r \rightarrow d_3}(A_1 \cap c^{-1}(r)) \cap W^3 = \{u_{11}\}$. By contracting the edges $(u_1, u_6)$, $(u_2, u_6)$, $(u_2, u_9)$, $(u_9, u_{19})$, $(u_{10}, u_{19})$, $(u_3, u_{10})$, all the vertices in $A_1$ are combined together. We also contract the edges $(u_4, u_{13})$ and $(u_5, u_{13})$ to obtain a tree isomorphic to $T^d_1$ in the subgraph $G_{c|Z_1}$. Let $E$ be the set of the aforementioned edges. Then, the resulting graph $E(G_c)$ contains a tree isomorphic to $T$. Hence, $W = \{u_1, u_2, u_3\}$. 