Exponential-Time Quantum Algorithms for Graph Coloring Problems

Kazuya Shimizu[1] · Ryuhei Mori[1]

Abstract

The fastest known classical algorithm deciding the k-colorability of n-vertex graph requires running time $\Omega(2^n)$ for $k \geq 5$. In this work, we present an exponential-space quantum algorithm computing the chromatic number with running time $O(1.9140^n)$ using quantum random access memory (QRAM). Our approach is based on Ambainis et al's quantum dynamic programming with applications of Grover's search to branching algorithms. We also present a polynomial-space quantum algorithm not using QRAM for the graph 20-coloring problem with running time $O(1.9575^n)$. For the polynomial-space quantum algorithm, we essentially develop $(4 - \epsilon)^n$-time *classical* algorithms that can be improved quadratically by Grover's search.

Keywords Quantum algorithm · Graph coloring · Grover's search · Dynamic programming

1 Introduction

Exhaustive search is believed to be (almost) the fastest classical algorithm for many NP-complete problems including SAT, hitting set problem, etc [8]. Grover's quantum search quadratically improves the running time of exhaustive search [15]. Hence, the best classical running time for many NP-complete problems can be quadratically improved by quantum algorithms. On the other hand, non-trivial faster classical algorithms are known for some NP-complete problems including the travelling salesman problem (TSP), the graph coloring problem, etc. For these problems, more complicated techniques, such as dynamic programming, arithmetic algorithm based on inclusion–exclusion principle, etc., are used in the fastest known classical algorithms.

✉ Ryuhei Mori
mori@c.titech.ac.jp

Kazuya Shimizu
shimizu.k.ap@m.titech.ac.jp

[1] School of Computing, Tokyo Institute of Technology, Meguro-ku, Tokyo, Japan

Table 1 $O(2^{d_k^* n})$-time quantum algorithms not using QRAM for the graph k-coloring problem

k	d_k^*	$2^{d_k^*}$	k	d_k^*	$2^{d_k^*}$	k	d_k^*	$2^{d_k^*}$
3	0.2051	1.1528	9	0.8041	1.7460	15	0.9488	1.9303
4	0.4039	1.3231	10	0.8298	1.7775	16	0.9488	1.9303
5	0.5553	1.4695	11	0.8298	1.7775	17	0.9488	1.9303
6	0.6099	1.5261	12	0.8676	1.8246	18	0.9536	1.9366
7	0.7234	1.6511	13	0.8874	1.8499	19	0.9690	1.9575
8	0.7299	1.6585	14	0.8938	1.8580	20	0.9691	1.9575

It is not obvious how to boost these classical algorithms by a quantum computer. Recently, Ambainis et al. showed a general idea of quantum dynamic programming using quantum random access memory (QRAM), and showed quantum speedup for many NP-complete and NP-hard problems including TSP, set cover, etc [1]. Ambainis et al.'s work gives a new general method for exact exponential-time quantum algorithms.

In this work, we present exact exponential-time quantum algorithms for the graph coloring problem. The fastest known classical algorithm computes the chromatic number of n-vertex graph with running time $\text{poly}(n)2^n$ on the random access memory (RAM) model. The main result of this work is the following theorem.

Theorem 1 *There is an exponential-space bounded-error quantum algorithm using QRAM for the chromatic number problem with running time[1] $O^*\big((2^{37/35}3^{3/7}5^{-9/70}7^{-5/28})^n\big) = O(1.9140^n)$.*

The quantum algorithm in Theorem 1 is based on Ambainis et al's quantum dynamic programming for TSP with applications of Grover's search to Byskov's algorithm enumerating all maximal independent sets (MISs) of fixed size [7]. Byskov's algorithm is not naive exhaustive search, but is a branching algorithm (also referred as Branch & Reduce), for which Grover's search can be applied [12]. While RAM is a widely accepted model of classical computation, QRAM is sometimes criticized due to the difficulty of implementation.
In this paper, we also present quantum algorithms not using QRAM.

Theorem 2 *For $k \leq 20$, there exists a constant $\epsilon > 0$ such that there are polynomial-space bounded-error quantum algorithms not using QRAM for the graph k-coloring problem with running time $2^{(1-\epsilon)n}$.*

Note that classical algorithms with running time $2^{(1-\epsilon)n}$ are known only for $k = 3, 4$ [2, 7]. Running times of the quantum algorithms in Theorem 2 are shown in Table 1. For proving Theorem 2, we essentially develop *classical* algorithms with running time $4^{(1-\epsilon)n}$ that can be improved quadratically by Grover's search. These classical algorithms are obtained by generalizing Byskov's techniques for reducing the graph $k(\geq 4)$-coloring problem to the graph 3-coloring problem [7]. The proofs

---

[1] In this paper, $O^*(f(n))$ means $O(\text{poly}(n)f(n))$.

of Theorems 1 and 2 in this paper require numerical calculations. More precisely, the running time is represented by the maximum of some complicated function. The maximum is approximately computed by evaluating the function at many points.

1.1 Related Work

Since a graph is k-colorable if and only if the set of vertices can be partitioned into k independent sets, many algorithms for the graph k-coloring problem use enumeration algorithms of independent sets. There is a simple branching algorithm enumerating all MISs in time $O^*(3^{n/3}) = O(1.4423^n)$ [11]. Lawler showed that 3-colorability can be decided in time $O^*(3^{n/3})$ by enumerating all MISs and checking the bipartiteness of the subgraph induced by the complement of each MIS [17]. Lawler also showed that the chromatic number can be computed in time $O(2.4423^n)$ by a simple dynamic programming.

Beigel and Eppstein showed an efficient algorithm for the graph 3-coloring problem with running time $O(1.3289^n)$ [2]. Byskov showed reduction algorithms from the graph $k(\geq 4)$-coloring problem to the graph 3-coloring problem [7]. By using Beigel and Eppstein's graph 3-coloring algorithm, Byskov showed classical algorithms for the graph 4-, 5- and 6-coloring problems with running time $O(1.7504^n)$, $O(2.1592^n)$ and $O(2.3289^n)$, respectively. Fomin et al. showed an algorithm for the graph 4-coloring problem with running time $O(1.7272^n)$ by using the path decomposition [10].

In 2006, Björklund and Husfeldt, and Koivisto showed an exponential-space $O^*(2^n)$-time algorithm for the chromatic number problem in the RAM model [3, 16]. These algorithms are based on the inclusion–exclusion principle. They also showed that if there is a polynomial-space $O^*(\alpha^n)$-time algorithm counting the number of independent sets, then there is a polynomial-space $O^*((1+\alpha)^n)$-time algorithm computing the chromatic number [3, 5]. Since the fastest known polynomial-space algorithm computes the number of independent sets with running time $O(1.2356^n)$ [13], there is a polynomial-space $O(2.2356^n)$-time algorithm computing the chromatic number.

There is almost no previous theoretical work on quantum algorithms for the graph coloring problems. Fürer mentioned that Grover's algorithm can be applied to branching algorithms, and hence, Beigel and Eppstein's algorithm for the graph 3-coloring problem can be improved to running time $O(\sqrt{1.3289}^n) = O(1.1528^n)$ [12]. The quantum algorithms for Theorem 2 are basically obtained by applying Grover's search to generalized Byskov's reduction algorithms on the basis of Fürer's observation.

For general NP-hard problems, Ambainis et al. showed exponential-space exponential-time quantum algorithms using QRAM for many NP-hard problems [1]. The quantum algorithm for Theorem 1 is based on Ambainis et al's algorithm for TSP with application of Grover's search to Byskov's algorithm enumerating MISs of fixed size on the basis of Fürer's observation.

1.2 Overview of Quantum Algorithms

1.2.1 Quantum Algorithm for the Chromatic Number Problem

Similarly to Ambainis et al.'s quantum algorithm for TSP, the quantum algorithm for Theorem 1 is a simple divide-and-conquer algorithm with dynamic programming approach. The basic classical algorithm was shown in [4,Proposition 3]. The chromatic number of a graph G is equal to a sum of the chromatic numbers of $G[S]$ and $G[V \setminus S]$ for some non-empty $S \subsetneq V$, unless G is one-colorable. If we can assume that S has size exactly $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$, then we can consider a classical algorithm that recursively finds S of size $\lfloor n/2 \rfloor$ minimizing $\chi(G[S]) + \chi(G[V \setminus S])$. Let $T(n)$ be the running time of this algorithm. Then, it follows $T(n) = \binom{n}{\lfloor n/2 \rfloor}(T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil))$, so that we can apply Ambainis et al's quantum dynamic programming straightforwardly and obtain an $O(1.7274^n)$-time quantum algorithm [1]. However, the balanced partition S satisfying $\chi(G) = \chi(G[S]) + \chi(G[V \setminus S])$ does not necessarily exist. Hence, we use the following useful fact.

Fact 1 *Let $a_1, \ldots, a_k$ be positive integers, and $n := \sum_{i=1}^{k} a_i$. Assume that $a_1 \geq a_i$ for all $i \in \{1, 2, \ldots, k\}$. Then, for any $m \in \{1, 2, \ldots, n-1\}$, there exists $S \subseteq \{2, 3, \ldots, k\}$ such that $\sum_{i \in S} a_i \leq m$ and $\sum_{i \in \{2, \ldots, k\} \setminus S} a_i \leq n - m - 1$.*

Proof Let $t := \max\{j \in \{2, \ldots, k\} \mid \sum_{i=2}^{j} \leq m\}$. Let $S := \{2, 3, \ldots, t\}$. Then, $\sum_{i \in \{t+1, t+2, \ldots, k\}} a_i \leq \sum_{i \in \{1, t+2, t+3, \ldots, k\}} a_i = n - \sum_{i \in \{2, 3, \ldots, t+1\}} a_i \leq n - m - 1$. $\square$

From Fact 1, we can consider the following quantum algorithm computing the chromatic number. First, the algorithm precomputes the chromatic number of all induced subgraphs with size at most $\lfloor n/4 \rfloor$. This precomputation is based on Lawler's formula

$$\chi(G) = 1 + \min_{I \in \text{MIS}(G)} \chi(G[V \setminus I]) \tag{1}$$

where $\text{MIS}(G)$ denotes the set of all MISs of G [17]. There is a classical algorithm enumerating all MISs with running time $O^*(3^{n/3})$. We will show in Sect. 3 that Grover's search can be applied to this algorithm, and hence, the quantum algorithm can search for all MISs with running time $O^*(3^{n/6})$. Here, computed chromatic numbers are stored to QRAM. Hence, we can apply Grover's search for computing the minimum in (1). The precomputation requires the running time $O^*\left(\sum_{i=1}^{\lfloor n/4 \rfloor} \binom{n}{i} 3^{i/6}\right) = O(1.8370^n)$. Then, the main part of the algorithm computes the chromatic number of G by using the formula

$$\chi(G) = 1 + \min_{I \in \text{MIS}(G)} \min_{S \subseteq V \setminus I, |S| \leq \lfloor n/2 \rfloor, |V \setminus I \setminus S| \leq \lfloor n/2 \rfloor} \left\{\chi(G[S]) + \chi(G[V \setminus I \setminus S])\right\}$$

for $\chi(G) \geq 3$. This formula is justified by Fact 1 for $m = \lfloor n/2 \rfloor$. Grover's search is used for finding S. For computing $\chi(G[S])$ and $\chi(G \setminus I \setminus S)$, the above formula is used again. Then, we need the chromatic numbers of subgraphs of G of size at most

$\lfloor n/4 \rfloor$, which were precomputed and stored to QRAM. The running time of the main part of this quantum algorithm is

$$O^* \left(3^{n/6} \sqrt{\binom{n}{n/2}} 3^{n/12} \sqrt{\binom{n/2}{n/4}} \right) = O(2.2134^n).$$

Quantum algorithm for Theorem 1 searches for all MISs of size t for each $t \in \{1, 2, \ldots, n\}$ separately. Therefore, the above estimate of the running time is larger than the actual running time since if MIS I of size t is chosen, the remaining graph $G[V \setminus I]$ has only $n - t$ vertices. Hence, the factor $2^{n/2}$ in the above estimate can be replaced by $2^{(n-t)/2}$. Then, precise analysis shows that the running time of the improved quantum algorithm is $O(1.9140^n)$.

1.2.2 Quantum Algorithms for the Graph k-coloring Problem

We will derive classical algorithms that can be improved quadratically by Grover's search. The ideas of the classical algorithms were obtained by Byskov [7]. In the classical algorithms, the graph $k(\geq 4)$-coloring problem is reduced to the graph k'-coloring problems for some $k' < k$. A graph G is $k(\geq 2)$-colorable if and only if there exists a subset S of vertices such that $G[S]$ is $\lfloor k/2 \rfloor$-colorable and $G[V \setminus S]$ is $\lceil k/2 \rceil$-colorable. Let us consider a classical algorithm that simply searches for $S \subseteq V$ satisfying the above condition. Let $T_k(n)$ be the running time of this algorithm for the graph k-coloring problem. Then, $T_k(n)$ satisfies

$$T_1(n) = T_2(n) = 1,$$
$$T_k(n) = \sum_{i=0}^{n} \binom{n}{i} (T_{\lfloor k/2 \rfloor}(i) + T_{\lceil k/2 \rceil}(n - i)), \quad k \geq 3$$

where polynomial factors in n are ignored. Then, we obtain $T_4(n) = O^*(2^n)$, $T_8(n) = O^*(3^n)$ and $T_{16}(n) = O^*(4^n)$. Let us consider a quantum algorithm that uses Grover's search for finding S. Let $T_k^*(n)$ be the running time of the quantum algorithm. Then, it follows $T_k^*(n) = \sum_{i=0}^{n} \sqrt{\binom{n}{i}} (T_{\lfloor k/2 \rfloor}^*(i) + T_{\lceil k/2 \rceil}^*(n - i))$, which implies $T_k^*(n) = O^*(\sqrt{T_k(n)})$. Hence, we obtain $T_4^*(n) = O(1.4143^n)$, $T_8^*(n) = O(1.7321^n)$ and $T_{16}^*(n) = O^*(2^n)$. This yields a weaker version of Theorem 2 that is valid for $k \leq 8$ rather than $k \leq 20$. Better reduction algorithms are used for Theorem 2.

1.3 Organization

In Sect. 2, notations and known classical and quantum algorithms are introduced. In Sect. 3, we present details of quantum algorithm for branching algorithms. In Sect. 4, we prove Theorem 1. In Sect. 5, we prove a weaker version of Theorem 2 that is valid for $k \leq 19$ rather than $k \leq 20$. Theorem 2 is obtained by improving the quantum algorithms in Sect. 5. The details of the proof of Theorem 2 are shown in Appendix B.

2 Preliminaries

2.1 Definitions and Notations

For a finite vertex set V, a set E of edges consists of subsets of V of size two. A pair (V, E) of finite vertex set V and a set E of edges is called an undirected simple graph. In this paper, we simply call a graph rather than an undirected simple graph. The number of vertices $|V|$ is denoted by n. A mapping $c\colon V \to \{1, 2, \ldots, k\}$ is called k-coloring if $c(v) \neq c(w)$ for all $\{v, w\} \in E$. For a graph G, the smallest k such that there exists a k-coloring is called the chromatic number of G, and denoted by $\chi(G)$. A subset $I \subseteq V$ of vertices is called an independent set if $\{v, w\} \notin E$ for all $v, w \in I$. An independent set I is said to be maximal if there is no strict superset of I that is an independent set. A maximal independent set of size t is called t-MIS. For $S \subseteq V$, $G[S]$ denotes the induced subgraph $(S, \{\{v, w\} \in E \mid v, w \in S\})$ of G. Let $h(\delta) := -\delta \log \delta - (1 - \delta) \log(1 - \delta)$ for $\delta \in [0, 1]$ where $0 \log 0 = 0$. In this paper, the base of logarithm is 2. The notation $g(n) = O^*(f(n))$ means that $g(n) = O(n^c f(n))$ for some constant c. For $O^*(\lambda^n)$, we often round λ up to the fourth digit after the decimal point. In this case, we can use $O()$ rather than $O^*()$. For example, we often write $g(n) = O(1.4143^n)$ rather than $g(n) = O^*(2^{n/2})$. The notation $g(n) = \widetilde{O}(f(n))$ means that $g(n) = O((\log f(n))^c f(n))$ for some constant c.

2.2 Known Classical Algorithm for Enumerating all t-MISs

Byskov showed the following theorem.

Theorem 3 (Byskov [7]) *The maximum number of t-MISs of n-vertex graphs is*

$$I(n, t) := \lfloor n/t \rfloor^{(\lfloor n/t \rfloor + 1)t - n} (\lfloor n/t \rfloor + 1)^{n - \lfloor n/t \rfloor t}.$$

Furthermore, there is a classical algorithm enumerating all t-MISs of an n-vertex graph in time $O^*(I(n, t))$.

We can straightforwardly obtain the following lemma and corollary.

Lemma 1 *For any constant $\delta \in (0, 1)$, $I(n, \lfloor \delta n \rfloor) = O(2^{E(\delta)n})$ where*

$$E(\delta) := ((\lfloor \delta^{-1} \rfloor + 1)\delta - 1) \log \lfloor \delta^{-1} \rfloor + (1 - \lfloor \delta^{-1} \rfloor \delta) \log(\lfloor \delta^{-1} \rfloor + 1).$$

Here, $E(\delta)$ is concave (and hence, continuous) and piecewise linear for $\delta \in (0, 1)$. The maximum of $E(\delta)$ is given at $\delta = 1/3$.

Proof It is easy to see that $I(n, \lfloor \delta n \rfloor) = O(2^{E(\delta)n})$. If $\lfloor \delta^{-1} \rfloor$ in the definition of $E(\delta)$ is replaced by δ^{-1}, we obtain $-\delta \log \delta$, which is obviously concave for $\delta \in (0, 1)$. Furthermore, $E(\delta) = -\delta \log \delta$ if δ is an inverse integer. For $\delta \in (1/(s + 1), 1/s)$ where s is some positive integer, $E(\delta)$ is a linear function since $\lfloor \delta^{-1} \rfloor$ is constant in this domain. Therefore, it is sufficient to show that $E(\delta)$ is continuous at $\delta = 1/s$ for

all $s \in \mathbb{Z}_{\geq 1}$ for showing that $E(\delta)$ is concave. Obviously, $E(\delta)$ is left-continuous. At $\delta = 1/s$ for some positive integer s, $E(\delta)$ is equal to $-\delta \log \delta = (\log s)/s$ even if $\lfloor \delta^{-1} \rfloor = s$ is replaced by $s - 1$. This means that $E(\delta)$ is right-continuous as well.

Finally, by comparing $E(1/2)$, $E(1/3)$ and $E(1/4)$, it is shown that the maximum of $E(\delta)$ is given at $\delta = 1/3$. $\qquad\square$

Corollary 1 *For any $a \in \mathbb{R}_{\geq 0}$ and $t \in \mathbb{Z}_{\geq 3}$, the maximum of $E(\delta) - a\delta$ for $\delta \in [1/t, 1]$ is given at $\delta = 1/s$ for some $s \in \{3, 4, \ldots, t\}$.*

2.3 Grover's Search

Here, Grover's search is briefly explained without introducing quantum circuit, unitary oracle, etc.

Theorem 4 (Grover [15], Boyer et al. [6]) *Let $A\colon \{1, 2, \ldots, N\} \to \{0, 1\}$ be a bounded-error quantum algorithm with running time T. Then, there is a bounded-error quantum algorithm computing $\bigvee_{x \in \{1, \ldots, N\}} A(x)$ with running time $\widetilde{O}(\sqrt{N}T)$. If it is guaranteed that either $|A^{-1}(1)| \geq M$ or $|A^{-1}(1)| = 0$, then there is a bounded-error quantum algorithm with running time $\widetilde{O}(\sqrt{N/M}T)$.*

Theorem 5 (Dürr and Høyer [9]) *Let $A\colon \{1, 2, \ldots, N\} \to \{1, 2, \ldots, M\}$ be a bounded-error quantum algorithm with running time T. Then, there is a bounded-error quantum algorithm computing $\min_{x \in \{1, \ldots, N\}} A(x)$ with running time $\widetilde{O}(\sqrt{N}T)$.*

2.4 QRAM

QRAM is the quantum analogue of RAM that can be accessed in a superposition [14]. QRAM has been used in many quantum algorithms [1]. RAM is the memory that can be accessed in constant or logarithmic time with respect to the memory size. For computing the minimum of some function $f(x, W)$ for all $x \in \{1, 2, \ldots, N\}$ where W denotes a read-only RAM, we can replace RAM with QRAM and apply Grover's search. Then, we obtain an $\widetilde{O}(\sqrt{N}T)$-time quantum algorithm computing $\min_x f(x, W)$ where T denotes the running time for evaluating f.

3 Grover's Search for Branching Algorithms

Fürer mentioned that Grover's search can be applied to branching algorithms [12]. Since the details of the quantum algorithm were not explicitly described in [12], we will show the details in this section. A branching algorithm is an algorithm which recursively reduce a problem into some problems of smaller parameters. We now consider decision problems with ℓ parameters $n_1, n_2, \ldots, n_\ell$ that are non-negative integers. If the parameters are sufficiently small, we do not apply any branching rule and solve this problem in some way. For a problem P with parameters $n_1, \ldots, n_\ell$ that are not sufficiently small, we choose a branching rule $b(P)$ such that P is reduced to $m_{b(P)}$ problems $P_1, P_2 \ldots, P_{m_{b(P)}}$ of the same class. Here, P_i has parameters

$f_1^{b(P),i}(n_1), \ldots, f_\ell^{b(P),i}(n_\ell)$ for some function $f_j^{b(P),i}$ satisfying $f_j^{b(P),i}(n_j) \leq n_j$ for $i = 1, 2, \ldots, m_{b(P)}$ and $j = 1, 2, \ldots, \ell$. At least one of the parameters of P_i must be smaller than the same parameter of P for all $i \in 1, 2, \ldots, m_{b(P)}$. The solution of P is true if and only if at least one of the solutions of $P_1, \ldots, P_{m_{b(P)}}$ is true. Hence, we will call this algorithm OR-branching algorithm. For a problem P of this class, we can consider a *computation tree* that represents the branchings of the reductions. The computation tree for P is a single node if P has sufficiently small parameters, so that no branching rule is performed, and is a rooted tree where children of the root node are the root nodes of the computation trees for $P_1, P_2, \ldots, P_{m_{b(P)}}$ if some branching rule $b(P)$ is applied to P. Let $L(n_1, \ldots, n_\ell)$ be the maximum number of leaves of the computation tree for P with parameters $n_1, \ldots, n_\ell$. Assume that the running time of the computation at a non-leaf node, including computations of $b(P)$, P_i, and $f_j^{b(P),i}$, is polynomial with respect to $n_1, \ldots, n_\ell$. Then, the total running time of the OR-branching algorithm is at most $L(n_1, \ldots, n_\ell)(\text{poly}(n_1, \ldots, n_\ell) + T)$ where T is the running time for the computation at a leaf node. We can apply Grover's search to OR-branching algorithms if we have an upper bound of $L(n_1, \ldots, n_\ell)$ with some properties, as shown in the following lemma.

Lemma 2 *Let $U(n_1, \ldots, n_\ell)$ be an upper bound of $L(n_1, \ldots, n_\ell)$ that can be computed in polynomial time with respect to the parameters, and satisfies*

$$U(n_1, \ldots, n_\ell) \geq \sum_{i=1}^{m_b} U(f_1^{b,i}(n_1), \ldots, f_\ell^{b,i}(n_\ell))$$

for any branching rule b. Then, there is a bounded-error quantum algorithm with running time $\sqrt{U(n_1, \ldots, n_\ell)}(\text{poly}(n_1, \ldots, n_\ell) + T)$.

Proof If we can assign an integer $s \in \{1, 2, \ldots, U(n_1, \ldots, n_\ell)\}$ to every leaf of the computation tree, and can compute the corresponding leaf from given s in polynomial time with respect to the parameters, then, we can apply Grover's search for computing

$$a(P) = \bigvee_{Q \in W(P)} a(Q)$$

where $a(P)$ denotes the solution of a problem P and $W(P)$ denotes the set of all problems corresponding to leaves of the computation tree for P. Then, we obtain a quantum algorithm with running time $\sqrt{U(n_1, \ldots, n_\ell)}(\text{poly}(n_1, \ldots, n_\ell) + T)$ as briefly mentioned in [12]. The algorithm computing the s-th leaf of a problem P is shown in Algorithm 1, which was omitted in [12]. We will show the validity of Algorithm 1.

Proposition 1 *For any problem Q that corresponds to a leaf node of the computation tree of a problem P with parameters $n_1, \ldots, n_\ell$, there exists $s \in \{1, 2, \ldots, U(n_1, \ldots, n_\ell)\}$ such that $\text{LEAF}(P, s) = Q$.*

Proof The proof is an induction on the depth of the computation tree for P. If the computation tree for P consists of a single node, then Algorithm 1 returns P for any s. Since

---

Algorithm 1 Algorithm computing s-th leaf of P

1: **function** LEAF(P, s)
2:     **if** P is a leaf **then return** P
3:     Compute the branching rule $b \leftarrow b(P)$
4:     **for** $i \in \{1, 2, \ldots, m_b - 1\}$ **do**
5:         Compute P_i and its parameters $n'_1, \ldots, n'_\ell = f_1^{b,i}(n_1), \ldots, f_\ell^{b,i}(n_\ell)$
6:         **if** $s \leq U(n'_1, \ldots, n'_\ell)$ **then return** LEAF(P_i, s)
7:         **else** $s \leftarrow s - U(n'_1, \ldots, n'_\ell)$
8:     **return** LEAF(P_{m_b}, s)

$U(n_1, \ldots, n_\ell)$ is an upper bound of $L(n_1, \ldots, n_\ell)$, it follows $U(n_1, \ldots, n_\ell) \geq 1$, and hence, $\{1, 2, \ldots, U(n_1, \ldots, n_\ell)\}$ is non-empty. Assume that the proposition holds for any P with the computation tree of depth at most d. We will consider a problem P with computation tree of depth $d+1$. Let i be the index of the branching at P that achieves Q. From the induction hypothesis, there exists $s' \in \{1, \ldots, U(f_1^{b,i}(n_1), \ldots, f_\ell^{b,i}(n_\ell))\}$ such that LEAF(P_i, s') $= Q$. Let $s := s' + \sum_{j=1}^{i-1} U(f_1^{b,j}(n_1), \ldots, f_\ell^{b,j}(n_\ell))$. Then, LEAF($P, s$) $= Q$. Here, $s \leq \sum_{j=1}^{m_b} U(f_1^{b,j}(n_1), \ldots, f_\ell^{b,j}(n_\ell)) \leq U(n_1, \ldots, n_\ell)$. □

From Proposition 1 and a fact that LEAF(P, s) always returns a problem corresponding to one of the leaf nodes for P, we obtain

$$a(P) = \bigvee_{s \in \{1, \ldots, U(n_1, \ldots, n_\ell)\}} a(\text{LEAF}(P, s)).$$

Since the depth of the computation tree for P is at most $\sum_{j=1}^{\ell} n_j$, the running time of LEAF(P, s) is polynomial with respect to the parameters. We have assumed that there is a quantum algorithm computing $a(Q)$ with running time T for $Q \in W(P)$. Hence, there is a quantum algorithm computing $a(\text{LEAF}(P, s))$ for given $s \in \{1, \ldots, U(n_1, \ldots, n_\ell)\}$ with running time poly$(n_1, \ldots, n_\ell) + T$. By using the quantum algorithm computing $a(\text{LEAF}(P, s))$ as the oracle, Grover's search computes $a(P)$ with running time $\sqrt{U(n_1, \ldots, n_\ell)}(\text{poly}(n_1, \ldots, n_\ell) + T)$. □

For a problem P whose solution is an integer, we can also consider a branching algorithm satisfying $a(P) = \min_{i=1}^{m_{b(P)}} a(P_i)$ for children $P_1, \ldots, P_{m_{b(p)}}$ of P. In this case, we will call this algorithm MIN-branching algorithm. Similarly to OR-branching algorithm, we can apply Grover's search to MIN-branching algorithm from Theorem 5.

In this paper, we apply Lemma 2 to Byskov's algorithm in Theorem 3. Byskov showed the upper bound $I(n, t)$ satisfying the conditions in Lemma 2 for the branching algorithm with two parameters n and t [7, Theorem 2]. Hence, we can apply Grover's search to Byskov's algorithm in Theorem 3. For example, since $\sum_{t=1}^{n} I(n, t) = O^*(3^{n/3})$, there is a bounded-error quantum algorithm searching for all MISs in time $O^*(3^{n/6})$.

4 Quantum Algorithms for the Chromatic Number Problem

The overview of the quantum algorithm was described in Sect. 1.2.1. The quantum algorithm for Theorem 1 is shown in Algorithm 2.

Algorithm 2 Algorithm computing the chromatic number of G. Grover's search is used for mins.

1: **function** CHR(G)
2:   **if** G is two colorable **then return** the chromatic number of G
3:   $\chi[\varnothing] \leftarrow 0$
4:   **for** $S \subseteq V$, $S \neq \varnothing$, $|S| \leq \lfloor n/4 \rfloor$ **do** (In the loop, S is earlier than S' if $S \subseteq S'$)
5:     $\chi[S] \leftarrow 1 + \min_{I \in \mathrm{MIS}(G[S])} \{\chi[S \setminus I]\}$
6:   **return** CHR1(V)

7: **function** CHR1(S)
8:   $c \leftarrow |S|$
9:   **for** $t \in \{1, \ldots, |S|\}$, $s \in \{\max\{\lceil |S|/2 \rceil - t, 1\}, \ldots, \lfloor (|S| - t)/2 \rfloor\}$ **do**
10:     $a \leftarrow \min_{I \in \mathrm{MIS}(G[S]), |I|=t} \min_{T \subseteq S \setminus I, |T|=s} (\mathrm{CHR2}(T) + \mathrm{CHR2}(S \setminus I \setminus T))$
11:     $c \leftarrow \min\{c, a\}$
12:   **return** $c + 1$

13: **function** CHR2(S)
14:   **if** $G[S]$ is two colorable **then return** the chromatic number of $G[S]$
15:   $c \leftarrow |S|$
16:   **for** $t \in \{1, \ldots, |S|\}$, $s \in \{\max\{\lceil |S|/2 \rceil - t, 1\}, \ldots, \lfloor (|S| - t)/2 \rfloor\}$ **do**
17:     $a \leftarrow \min_{I \in \mathrm{MIS}(G[S]), |I|=t} \min_{T \subseteq S \setminus I, |T|=s} (\chi[T] + \chi[S \setminus I \setminus T])$
18:     $c \leftarrow \min\{c, a\}$
19:   **return** $c + 1$

For computing the chromatic number of $G[S]$, when MIS I of size t is chosen, we have to chose $T \subseteq S \setminus I$ satisfying $|T| \leq |S|/2$ and $|S \setminus I \setminus T| \leq |S|/2$ as mentioned in Sect. 1.2.1. This implies the condition $|S|/2 - t \leq |T| \leq |S|/2$. Hence, Algorithm 2 computes the chromatic number correctly. By analyzing the running time of Algorithm 2, we obtain the following theorem.

Theorem 6 *Algorithm 2 computes the chromatic number of an n-vertex graph with running time* $O^* \big((2^{37/35} 3^{3/7} 5^{-9/70} 7^{-5/28})^n \big) = O(1.9140^n)$ *with bounded error probability.*

Proof The running time of the precomputation is $O^* \left(\sum_{i=1}^{\lfloor n/4 \rfloor} \binom{n}{i} 3^{i/6} \right) = O^* \big(2^{h(1/4)n} 3^{n/24} \big) = O(1.8370^n)$. Let $T_1(n)$ be the running time of CHR1(V) and $T_2(m)$ be the running time of CHR2(S) for $S \subseteq V$ of size m. Then, we obtain

$$T_2(m) = \sum_{t=1}^{m} \sqrt{I(m, t)} \sum_{s=\max\{\lceil m/2 \rceil - t, 1\}}^{\lfloor (m-t)/2 \rfloor} \sqrt{\binom{m-t}{s}}, \tag{2}$$

$$T_1(n) = \sum_{t=1}^{n} \sqrt{I(n, t)} \sum_{s=\max\{\lceil n/2 \rceil - t, 1\}}^{\lfloor (n-t)/2 \rfloor} \sqrt{\binom{n-t}{s}} (T_2(s) + T_2(n - t - s))$$

$$\leq \sum_{t=1}^{n} \sqrt{I(n,t)} \sum_{s=0}^{\min\{\lfloor n/2\rfloor, n-t\}} \sqrt{\binom{n-t}{s}} T_2(s) \tag{3}$$

by ignoring polynomial factors in n. Here, $T_2(m) \leq \sum_{t=1}^{m} \sqrt{I(m,t)} 2^{\frac{m-t}{2}}$ whose exponent is equal to $\max_{\delta \in [0,1]} \{(E(\delta) + (1-\delta))/2\}$. From Corollary 1, it is sufficient to take maximum among δ being an inverse integer. Numerical calculation shows that the maximum is given at $\delta = 1/5$ and hence $T_2(m) = O^*(80^{m/10}) = O(1.5500^m)$. Hence, the exponent of $T_1(n)$ is equal to

$$\max_{\delta \in [0,1/3], \lambda \in [0,1/2]} \left\{ \frac{1}{2} E(\delta) + \frac{1}{2} h\left(\frac{\lambda}{1-\delta}\right)(1-\delta) + \left(\frac{1}{10}\log 80\right)\lambda \right\}.$$

Here, we only consider maximum for $t \leq n/3$ since $I(n,t)$ is decreasing with respect to t for $t \geq n/3$, and the other part $\sum_s \sqrt{\binom{n-t}{s}} T_2(s)$ in (3) is also decreasing with respect to t. Numerical calculation shows that the maximum is given at $\delta = 1/7, \lambda = 1/2$. Hence, we obtain

$$T_1(n) = O^* \left(\left(7^{1/14} 2^{h(7/12)3/7} 80^{1/20}\right)^n \right)$$
$$= O^* \left((2^{37/35} 3^{3/7} 5^{-9/70} 7^{-5/28})^n \right) = O(1.9140^n).$$

$\square$

Careful readers may notice that the running times of the precomputation and the main computation are not balanced. If the quantum algorithm precomputes the chromatic numbers of induced subgraphs with size at most $(1/4+\epsilon)n$ for some $\epsilon > 0$, the precomputation and the main computation require more and less running time, respectively (we can use Fact 1 for unbalanced m). At first glance, it might be possible to improve the running time by optimizing ϵ such that the both running time are balanced. Indeed, this idea improved the running time of the quantum algorithm for TSP [1], but does not improve the running time of Algorithm 2. Equation (2) is dominated by $t = m/5$ and $s = (2/5)m$. Equation (3) is dominated by $t = n/7$ and $s = n/2$. In order to exclude $s = (2/5)n$ in the summation in (2), the chromatic number of induced subgraph with size at most $(3/10)n$ must be precomputed. However, the running time of the precomputation in this case is $\sum_{i=1}^{(3/10)n} \binom{n}{i} 3^{i/6} = \Omega(1.9460^n)$. Hence, the running time of quantum algorithm is not improved.

5 Quantum Algorithms Not Using QRAM

5.1 Known Classical Algorithms for k-coloring Problem

Beigel and Eppstein showed the fastest known classical algorithm for the graph 3-coloring problem.

Theorem 7 (Beigel and Eppstein [2]) *There is a classical algorithm for the graph 3-coloring problem with running time* $O^*((2^{3/49}3^{4/49}\Lambda^{24/49})^n) = O(1.3289^n)$ *where* Λ *denotes the unique real positive root of* $x^5 - 2x - 2$.

For larger k, Byskov showed reduction algorithms from the graph k-coloring problem to the graph 3-coloring problem [7]. Since a graph G is k-colorable if and only if there exists an MIS I of size at least $\lceil n/k \rceil$ such that $G[V \setminus I]$ is $(k-1)$-colorable, we obtain the following reduction algorithm.

Reduction algorithm 1 (Byskov [7], Lawler [17]) *For each* $t \in \{\lceil n/k \rceil, \lceil n/k \rceil + 1, \ldots, n\}$, *enumerate all* t-*MISs. For each* t-*MIS* I, *the algorithm for the graph* $(k-1)$-*coloring is performed to* $G[V \setminus I]$.

Let $T_k^{(1)}(n)$ be the running time of an algorithm for the graph k-coloring problem using Reduction Algorithm 1. Then, it satisfies

$$T_k^{(1)}(n) = \sum_{t=\lceil n/k \rceil}^{n} I(n,t) T_{k-1}^{(1)}(n-t) \tag{4}$$

for $k \geq 4$ by ignoring a polynomial factor. By using Reduction Algorithm 1 and Theorem 7, Byskov obtained algorithms for the graph 4- and 5-coloring problems with running time $O(1.7504^n)$ and $O(2.1592^n)$, respectively. Byskov also introduced another reduction algorithm. Here, we introduce it in a general form. Since a graph G is k-colorable if and only if there exists a subset S of vertices of size at least $\lceil nk'/k \rceil$ such that $G[S]$ is k'-colorable and $G[V \setminus S]$ is $(k-k')$-colorable for arbitrary $k' < k$, we obtain the following reduction algorithm.

Reduction algorithm 2 (Byskov [7], Lawler [17]) *Fix* $k' \in \{2, 3, \ldots, \lfloor k/2 \rfloor\}$. *For each* $t \in \{\lceil nk'/k \rceil, \lceil nk'/k \rceil + 1, \ldots, n\}$, *enumerate all subsets of vertices of size* t. *For each subset* S *of vertices of size* t, *the algorithms for the graph* k'- *and* $(k-k')$-*coloring problems are performed to* $G[S]$ *and* $G[V \setminus S]$, *respectively.*

Let $T_k^{(2)}(n)$ be the running time of an algorithm for the graph k-coloring problem using Reduction Algorithm 2. Then, it satisfies

$$T_k^{(2)}(n) = \sum_{t=\lceil nk'/k \rceil}^{n} \binom{n}{t} \left(T_{k'}^{(2)}(t) + T_{k-k'}^{(2)}(n-t) \right) \tag{5}$$

for $k \geq 4$ by ignoring a polynomial factor. Reduction Algorithm 2 is a simple generalization of a reduction algorithm in [7]. For $k = 6$, Reduction Algorithm 2 with $k' = 3$ gives $T_6^{(2)}(n) = O(2.3289^n)$ while $T_6^{(1)}(n) = O(2.5602^n)$ [7]. Byskov did not consider classical algorithms for the graph $k(\geq 7)$-coloring problem since the classical algorithms obtained by Reduction Algorithms 1 and 2 are slower than the fastest classical algorithm computing the chromatic number at that time with running time $O(2.4023^n)$ [7].

5.2 Quantum Algorithms Not Using QRAM

In this section, we present quantum algorithms not using QRAM, and prove a weaker version of Theorem 2 that is valid for $k \leq 19$ rather than $k \leq 20$. Theorem 2 is obtained by improving quantum algorithms presented in this section. The improved quantum algorithm and the proof of Theorem 2 are shown in Appendix B.

Fürer mentioned that Grover's search can be applied to Beigel and Eppstein's algorithm.

Lemma 3 (Fürer [12]) *There is a bounded-error quantum algorithm not using QRAM for the graph 3-coloring problem with running time* $O^*((2^{3/49}3^{4/49}\Lambda^{24/49})^{n/2}) = O(1.1528^n)$ *where* Λ *denotes the unique real positive root of* $x^5 - 2x - 2$.

The quantum algorithm in Lemma 3 is obtained by application of Lemma 2 to classical Beigel and Eppstein's algorithm in Theorem 7. On the other hand, Beigel and Eppstein also showed a very simple randomized algorithm for the graph 3-coloring problem with running time $O^*(2^{n/2})$ [2,Corollary 1]. This randomized algorithm searches for one of the $2^{n/2}$ solutions from 2^n leaves in a quaternary computation tree of depth $n/2$. From Theorem 4, we can apply Grover's search to the randomized algorithm and obtain a quantum algorithm with running time $O^*(2^{n/4})$. The quantum algorithms for Theorem 2 reduce the graph k-coloring problems to the graph 3-coloring problem. Theorem 2 can be obtained even if this simpler quantum 3-coloring algorithm is used in place of the involved quantum 3-coloring algorithm in Lemma 3 although the exponents increase. By applying Reduction Algorithms 1 and 2 boosted by Grover's search, the following Theorem is obtained.

Theorem 8 *Assume that there is a polynomial-space bounded-error quantum algorithm not using QRAM for the graph 3-coloring problem with running time* $O^*(2^{f_3^* n})$ *for some* f_3^*. *Then, there is a polynomial-space bounded-error quantum algorithm not using QRAM for the graph k-coloring problem with running time* $O^*(2^{f_k^* n})$ *where*

$$f_k^* := \min \begin{cases} \max_{s \in \{3,4,\ldots,k\}}\{(\log s)/(2s) + (1 - 1/s)f_{k-1}^*\}, \\ \min_{2 \leq k' \leq \lfloor k/2 \rfloor} \max_{\delta \in [k'/k, 1]} \left\{ h(\delta)/2 + \max\{\delta f_{k'}^*, (1 - \delta)f_{k-k'}^*\} \right\} \end{cases}$$

for $k \geq 4$.

Proof We consider a quantum algorithm using Reduction Algorithms 1 and 2 with Grover's search. Since we can apply Lemma 2 to Byskov's enumeration algorithm of t-MIS in Reduction Algorithm 1, we obtain

$$T_k^*(n) \leq \sum_{t = \lceil n/k \rceil}^{n} \sqrt{I(n,t)} T_{k-1}^*(n-t).$$

For Reduction Algorithm 2, we can simply apply Grover's search, and obtain

$$T_k^*(n) \leq \sum_{t = \lceil nk'/k \rceil}^{n} \sqrt{\binom{n}{t}} \left(T_{k'}^*(t) + T_{k-k'}^*(n-t) \right)$$

for any $k' \in \{2, \ldots, \lfloor k/2 \rfloor\}$. Hence, by choosing the best reduction algorithm, we obtain quantum algorithms with running time $O^*(2^{F_k^* n})$ where $F_3^* = f_3^*$ and

$$F_k^* := \min \begin{cases} \max_{\delta \in [1/k, 1]} \left\{ E(\delta)/2 + (1 - \delta)F_{k-1}^* \right\}, \\ \min_{2 \le k' \le \lfloor k/2 \rfloor} \max_{\delta \in [k'/k, 1]} \left\{ h(\delta)/2 + \max\{\delta F_{k'}^*, (1 - \delta)F_{k-k'}^*\} \right\} \end{cases}$$

for $k \ge 4$. Since $E(\delta)$ is decreasing for $\delta \ge 1/3$, we can assume that $\delta \le 1/3$. From Corollary 1, it is sufficient to take maximum among $\delta = 1/s$ for $s \in \{3, 4, \ldots, k\}$. This proves $F_k^* = f_k^*$. □

By using Theorem 8 with $f_3^* = (3 + 4 \log 3 + 24 \log \Lambda)/98 \le 0.2051$ or $f_3^* = 1/4$, we obtain Theorem 2 for $k \le 19$. In Table 2 in Appendix A, the values of f_k^* and best choices of k' are shown for $f_3^* = (3 + 4 \log 3 + 24 \log \Lambda)/98$. Here, we summarize the quantum algorithm.

Algorithm 3 Algorithm for the k-colorability of G. Grover's search is used for two $\bigvee$s.

1: **function** COL(G, k)
2:    **if** $k \le 2$ **then return** the k-colorability of G by a polynomial-time algorithm
3:    **if** $k = 3$ **then return** the 3-colorability of G by Beigel and Eppstein's algorithm with Grover's search
4:    **if** $k \le 5$ **then**
5:       **for** $t \in \{\lceil n/k \rceil, \ldots, n\}$ **do**
6:          **if** $\bigvee_{I \in \mathrm{MIS}(G), |I| = t}$ COL($G[V \setminus I], k - 1$) **then return** true
7:       **return** false
8:    Choose k' depending on k from Table 2
9:    **for** $t \in \{\lceil nk'/k \rceil, \ldots, n\}$ **do**
10:       **if** $\bigvee_{S \subseteq V, |S| = t}$ COL($G[S], k'$) $\wedge$ COL($G[V \setminus S], k - k'$) **then return** true
11:    **return** false

It is easy to calculate the exponents f_k^* efficiently and precisely. The details are explained in Appendix A. Note that if Grover's search is not used in Algorithm 3, we obtain classical algorithms with running time $O^*(4^{f_k^* n})$. We will obtain Theorem 2 by improving Algorithm 3. When Reduction Algorithm 2 is applied, we can assume that $k - k'$ independent sets in $G[V \setminus S]$ are smaller than k' independent sets in $G[S]$. Hence, we can use the average size $|S|/k'$ of k' independent sets in $G[S]$ as an upper bound on the size of independent sets in $G[V \setminus S]$. This idea reduces the running time of Algorithm 3 and gives Theorem 2. The details are shown in Appendix B.

A Calculations of the Exponents f_k^*

Here, we consider calculations of the exponents f_k^*. The details of the numerical calculations were not written in [1]. Fortunately, the calculation of f_k^* is not difficult. The non-trivial part is the calculation of

$$\max\left\{\max_{\delta\in[k'/k,1]}\{h(\delta)/2+\delta f_{k'}^*\},\ \max_{\delta\in[k'/k,1]}\{h(\delta)/2+(1-\delta)f_{k-k'}^*\}\right\}.$$

In general, the sum of the binary entropy function and a linear function is a concave function. Hence, it is sufficient to find a stationary point δ^*. For maximizing $h(\delta)/2 + \delta f_{k'}^*$, we need to find $\delta^* \in [0, 1]$ such that

$$\frac{1}{2}\log\frac{1-\delta^*}{\delta^*} + f_{k'}^* = 0.$$

Since the left-hand side is monotonically decreasing, δ^* can be approximated efficiently by the binary search. If $\delta^* < k'/k$, then $\delta = k'/k$ gives the maximum. The precise values of f_k^* computed by the above algorithm and chosen k' are shown in Table 2. Here, $k' = 1$ means that Reduction Algorithm 1 is chosen.

Table 2 Precise values of f_k^*

k	f_k^*	$2^{f_k^*}$	k'	k	f_k^*	$2^{f_k^*}$	k'
3	0.2050919796	1.1527598391		12	0.8675130685	1.8245150716	6
4	0.4038189847	1.3230054317	1	13	0.8873694503	1.8498001987	6
5	0.5552479972	1.4694212030	1	14	0.9096459955	1.8785844800	6
6	0.6098104848	1.5260587298	3	15	0.9487955413	1.9302604739	7
7	0.7233677736	1.6510316464	3	16	0.9487955413	1.9302604739	7
8	0.7298058730	1.6584159226	4	17	0.9535113456	1.9365803294	8
9	0.8040091395	1.7459462428	4	18	0.9565265484	1.9406319746	8
10	0.8297793332	1.7774134780	5	19	0.9713689548	1.9607001959	8
11	0.8297793332	1.7774134780	5	20	1.0059831384	2.0083116140	8

B Improved Quantum Algorithm for *k*-coloring Problems Not Using QRAM

Theorem 9 *Assume that there is a polynomial-space bounded-error quantum algorithm not using QRAM for the graph 3-coloring problem with running time $O^*(2^{f_3^* n})$ for some f_3^*. Then, there is a polynomial-space bounded-error quantum algorithm not using QRAM for the graph k-coloring problem with running time $O^*(2^{d_k^*(1)n})$ where $d_3^*(\mu) := f_3^*$ for all $\mu \in [1/3, 1]$ and*

$$d_k^*(\mu) := \min \begin{cases} \max_{s \in \{2,3,\ldots,k\}} \big\{ (\log s)/(2s) + (1 - 1/s)d_{k-1}^*(1) \big\}, \\ \min_{2 \le k' \le \lfloor k/2 \rfloor} \max_{\delta \in [k'/k, \min\{1, \mu k'\}]} \big\{ h(\delta)/2 \\ \quad + \max\{\delta d_{k'}^*(\min\{1, \mu/\delta\}), \ (1-\delta)d_{k-k'}^*(\min\{1, \delta/(k'(1-\delta))\})\} \big\} \end{cases}$$

for $k \ge 4$ and $\mu \in [1/k, 1]$.

Proof The quantum algorithms are almost the same as those in Theorem 8. When we apply Reduction Algorithm 2, we check for all subsets S of vertices of size t whether $G[S]$ is k'-colorable and $G[V \setminus S]$ is $(k-k')$-colorable. Since we assume that S consists of k' largest independent sets in a coloring, we can safely assume that $|S| \ge \lceil nk'/k \rceil$. At the same time, we can assume that $k - k'$ independent sets in a coloring of $G[V \setminus S]$ have size at most $\lfloor |S|/k' \rfloor$, which is the average size of independent sets in a coloring of $G[S]$. This knowledge can be used for reducing the running time of quantum algorithms. We can consider a partial function that outputs true if the given graph can be partitioned into k independent sets of size at most u, outputs false if the given graph is not k-colorable, and outputs either of true or false for other cases. For computing this partial function, we can restrict the size of S in Reduction Algorithm 2. Since we only have to consider independent sets of size at most u, we can assume that $|S|$ is at most uk' and at least $n - u(k - k')$. We can assume that u is at least $\lceil n/k \rceil$ since otherwise there is no solution. Then, $\lceil nk'/k \rceil \ge n - u(k - k')$. Hence, we can assume that $|S|$ is at least $\lceil nk'/k \rceil$ and at most $\min\{n, uk'\}$. Let $T_k^*(n, u)$ denote the running time of the quantum algorithm. Then, we obtain

$$T_k^*(n, u) \le \sum_{t=\lceil n/k \rceil}^{n} \sqrt{I(n, t)} T_{k-1}^*(n - t, n - t),$$

$$T_k^*(n, u) \le \sum_{t=\lceil nk'/k \rceil}^{\min\{n, uk'\}} \sqrt{\binom{n}{t}} \left(T_{k'}^*(t, \min\{t, u\}) + T_{k-k'}^*(n - t, \min\{n - t, \lfloor t/k' \rfloor\}) \right)$$

for any $k' \in \{2, 3, \ldots, k\}$. Note that in Reduction Algorithm 1, we cannot assume that enumerated MIS has size at most u since the size of MIS is not restricted. On the other hand, $T_{k-1}^*(n-t, n-t)$ in the first inequality can be replaced by $T_{k-1}^*(n-t, \min\{t, u\})$. However, we do not apply this improvement since numerical calculation show that this does not improve the exponents of running time for $k \in \{3, 4, \ldots, 21\} \setminus \{13\}$. By choosing the best reduction algorithms, we obtain $T_k^*(n, \lfloor \mu n \rfloor) = O^*(2^{d_k^*(\mu)n})$. □

By using Theorem 9 with $f_3^* = (3 + 4 \log 3 + 24 \log \Lambda)/98$ or $1/4$, we obtain Theorem 2. Table 1 shows the values of $d_k^*(1)$. Here, k' may depend on u. However, even if k' is determined only by k, the same exponents are obtained. Chosen k's are the same as those in Table 2 except for $k' = 7$ for $k = 17$. Here, we summarize the quantum algorithm.

Algorithm 4 Algorithm for the k-colorability of G. If G can be partitioned into k independent sets of size at most u, then return true. If G is not k-colorable, then return false. Otherwise, return either of true or false. Grover's search is used for the two $\bigvee$s

---

1: **function** COL(G, k, u)
2:   **if** $k \leq 2$ **then return** the k-colorability of G by a polynomial-time algorithm
3:   **if** $k = 3$ **then return** the 3-colorability of G by Beigel and Eppstein's algorithm with Grover's search
4:   **if** $k \leq 5$ **then**
5:     **for** $t \in \{\lceil n/k \rceil, \ldots, n\}$ **do**
6:       **if** $\bigvee_{I \in \text{MIS}(G), |I|=t}$ COL($G[V \setminus I], k-1, |V \setminus I|$) **then return** true
7:     **return** false
8:   Choose k' that depends on k from Table 2, but $k' = 7$ for $k = 17$
9:   **for** $t \in \{\lceil nk'/k \rceil, \ldots, \min\{n, uk'\}\}$ **do**
10:     **if** $\bigvee_{S \subseteq V, |S|=t}$ COL($G[S], k', u$) $\wedge$ COL($G[V \setminus S], k-k', \lfloor t/k' \rfloor$) **then return** true
11:   **return** false

The details of numerical calculation of $d_k^*(1)$ and their precise values are shown in Appendix C. Finally, we introduce our ideas that failed to improve the running time. Similarly to Theorem 9, we can assume that in Reduction Algorithm 2, the size of independent sets in a coloring of $G[S]$ is lower bounded by $(n-t)/(k-k')$, which is the average size of independent sets in $G[V \setminus S]$. However, this idea could not improve the exponents in our numerical calculations. We also tried to use hybrid algorithms of Reduction Algorithms 1 and 2. We introduce a threshold s of size of independent sets. Then, Reduction Algorithms 1 and 2 are both applied on the assumptions that the size of the largest independent set in a coloring is at least $s+1$ and at most s, respectively. The threshold s is optimized so that the running time of Reduction algorithms are balanced. This algorithm failed to improve the running time as well.

C Calculations of the Exponents $d_k^*(\mu)$

It is much more difficult to calculate $d_k^*(1)$ than f_k^*. The non-trivial part is the calculations of

$$\max\left\{\max_{\delta \in [k'/k, \min\{1, \mu k'\}]} \{h(\delta)/2 + \delta d_{k'}^*(\min\{1, \mu/\delta\})\},\right.$$

$$\left.\max_{\delta \in [k'/k, \min\{1, \mu k'\}]} \{h(\delta)/2 + (1-\delta)d_{k-k'}^*(\min\{1, \delta/(k'(1-\delta))\})\}\right\}.$$

In the numerical calculations, the maximum for δ is taken for all $\delta \in \{1/2^{16}, 2/2^{16}, \ldots, 2^{16}/2^{16}\}$. Computed $d_k^*(\mu)$ is cached and reused. Then, we obtain $d_k^*(1)$ in Table 1.

Table 3 Precise values of $d_k^*(1)$. For $k \leq 13$, $d_k^*(1) = f_k^*$

k	$d_k^*(1)$	$2^{d_k^*(1)}$	k'
13	0.8873694503	1.8498001987	6
14	0.8937052065	1.8579416667	6
15	0.9487955413	1.9302604739	7
16	0.9487955413	1.9302604739	7
17	0.9487955413	1.9302604739	7
18	0.9535113456	1.9365803294	8
19	0.9689936620	1.9574747012	8
20	0.9690025400	1.9574867472	8
21	1.0086631422	2.0120457954	9

As another heuristic way, we assume that for each k', the functions $h(\delta)/2 + \delta d_{k'}^*(\min\{1, \mu/\delta\})$ and $h(\delta)/2 + (1 - \delta)d_{k-k'}^*(\min\{1, \delta/(k'(1 - \delta))\})$ are unimodal, i.e., functions with a single maximal. On this assumption, we can apply the golden-section search for finding the maximum efficiently. Obtained approximations of $d_k^*(1)$ are very close to those calculated by the first method. Hence, we believe that the golden-section search gives very precise approximations of $d_k^*(1)$, which are shown in Table 3.

References

1. Ambainis, A., Balodis, K., Iraids, J., Kokainis, M., Prūsis, K., Vihrovs, J.: Quantum speedups for exponential-time dynamic programming algorithms. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19), pp. 1783–1793. SIAM (2019)
2. Beigel, R., Eppstein, D.: 3-coloring in time $O(1.3289^n)$. J. Algorithms **54**(2), 168–204 (2005)
3. Björklund, A., Husfeldt, T.: Inclusion-exclusion algorithms for counting set partitions. In: Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp. 575–582. IEEE (2006)
4. Björklund, A., Husfeldt, T.: Exact algorithms for exact satisfiability and number of perfect matchings. Algorithmica **52**(2), 226–249 (2008)
5. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM J. Comput. **39**(2), 546–563 (2009)
6. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik: Prog. Phys. **46**(4–5), 493–505 (1998)
7. Byskov, J.M.: Enumerating maximal independent sets with applications to graph colouring. Oper. Res. Lett. **32**(6), 547–556 (2004)
8. Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlström, M.: On problems as hard as CNF-SAT. ACM Trans. Algorithms **12**(3), 41 (2016)
9. Dürr, C., Høyer, P.: A quantum algorithm for finding the minimum. arXiv preprint arXiv:quant-ph/9607014 (1996)
10. Fomin, F.V., Gaspers, S., Saurabh, S.: Improved exact algorithms for counting 3-and 4-colorings. In: Proceedings of the Thirteenth Annual International Computing and Combinatorics Conference (COCOON'07), pp. 65–74. Springer (2007)
11. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Springer, Berlin (2010)
12. Fürer, M.: Solving NP-complete problems with quantum search. In: Latin American Symposium on Theoretical Informatics (LATIN'08), pp. 784–792. Springer (2008)

13. Gaspers, S., Lee, E.J.: Faster graph coloring in polynomial space. In: Proceedings of the Twenty-third Annual International Computing and Combinatorics Conference (COCOON'17), pp. 371–383. Springer (2017)
14. Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. Phys. Rev. Lett. **100**(16), 160501 (2008)
15. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing (STOC'96), pp. 212–219. ACM (1996)
16. Koivisto, M.: An $O^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In: Proceedings of the Forty-seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp. 583–590. IEEE (2006)
17. Lawler, E.L.: A note on the complexity of the chromatic number problem. Inf. Proc. Lett. **5**, 66–67 (1976)